TRANSMITTAL OF APPEAL BRIEF (Large Entity)

Docket No. 90041.97R094/CSD

In Re Application Of: **R. Keith Riffee**

Serial No.	Filing Date	Examiner	Group Art Unit
08/800,574	February 18, 1997	Richard J. Lee	2613

Invention: **NARROWBAND VIDEO CODEC**

OIPE JC67 MAY 30 2003 PATENT & TRADEMARK OFFICE

TO THE COMMISSIONER FOR PATENTS:

Transmitted herewith in triplicate is the Appeal Brief in this application, with respect to the Notice of Appeal filed on

The fee for filing this Appeal Brief is: **$0.00** (PREVIOUSLY PAID)

☐ A check in the amount of the fee is enclosed.

☐ The Director has already been authorized to charge fees in this application to a Deposit Account.

☐ The Director is hereby authorized to charge any fees which may be required, or credit any overpayment to Deposit Account No.

[illegible]
Signature

Dated: 5/28/03

Thomas R. FitzGerald, Esquire
Reg. No. 26,730
16 E. Main Street, Suite 210
Rochester, New York 14614
Telephone: (585) 454-2250
Fax: (585) 454-6364

I certify that this document and fee is being deposited on May 28, 2003 with the U.S. Postal Service as first class mail under 37 C.F.R. 1.8 and is addressed to the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

Penny P. Clements
Signature of Person Mailing Correspondence

Penny P. Clements
Typed or Printed Name of Person Mailing Correspondence

cc:

CERTIFICATE OF MAILING BY FIRST CLASS MAIL (37 CFR 1.8)	Docket No.
Applicant(s): R. Keith Riffee	90041.97R094/CSD-55

Serial No.	Filing Date	Examiner	Group Art Unit
08/800,574	February 18, 1997	Richard J. Lee	2613

Invention: NARROWBAND VIDEO CODEC

OIPE JC67
MAY 3 0 2003
PATENT & TRADEMARK OFFICE

I hereby certify that this Appeal Brief (Revised)(in triplicate/ total 42 pages)
(Identify type of correspondence)

is being deposited with the United States Postal Service as first class mail in an envelope addressed to:

Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on May 29, 2003
(Date)

Penny P. Clements
(Typed or Printed Name of Person Mailing Correspondence)

Penny P. Clements
(Signature of Person Mailing Correspondence)

Note: Each paper must have its own certificate of mailing.

P07A/REV03

Docket:
90041.97R074 (CSD-55)

IN THE UNITED STATES PATENT & TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Applicant:	Robert K. Riffee	)	Examiner: Lee, Richard J.
Serial No.:	08/800,574	)	
Filed:	February 18, 1997	)	Art Unit: 2613
For:	NARROWBAND VIDEO CODEC	)	

APPEAL BRIEF (REVISED)

In response to the office action mailed 12/31/02, Applicant submits this Appeal Brief (Revised) in triplicate.

1. Real Party in Interest

The Real Party in Interest is Harris Corporation, having a place of business at 1025 West NASA Boulevard, Melbourne, Florida, 32919.

2. Related Appeals and Interferences

There are no related appeals or interferences.

3. Status of Claims

Claims 1-30 are pending, all claims are rejected and all claims are appealed.

4. Status of Amendments

There are no amendments after final.

5. Summary of Invention

Modern military operations now require real time two way digital video communications. A video codec (coder, decoder) converts video images into digital signals and vice versa. Present video codecs are large, power hungry, and complex. Such codecs are used at video conferencing facilities and are typically housed in a controlled environment of an office building. Large and complex devices are both inappropriate and virtually useless for military operations, especially combat operations. Accordingly, there is a need for a compact and efficient codec to transmit real time color video with audio over existing tactical communication links, in particular radio frequency (rf) links.

The invention provides a solution to the problem by providing a tactical military narrowband video codec 10 that transmits real time color video and audio over rf communication links. The narrowband video codec has a small chassis, is battery powered, and performs real time video and audio compression and decompression, forward error correction, and digital data transmission via tactical radios. The codec operates in a half-duplex or full-duplex mode and interfaces with a variety of standard video and audio signals.

The narrowband video codec 10 (Fig. 1) generates an output stream of control, data, and error correction bits. The codec frames the stream of bits into a series of sequential frames of bytes for transmission over an RF link of a controlled frequency. Each frame (Figs. 7a-7d) comprises an identical sequence of bytes and includes, in sequence, two control bytes, a plurality of sequential sets of data bytes and a plurality of error correction bytes. The data bytes include repeated sets of audio and video bytes. Each set of the data bytes has the audio and video bytes in the same order as each other set of data bytes in the frame. In particular, each set of data bytes has the same number of video bytes between sequential audio bytes. Each byte includes eight bits of data. The Frame Structure is described in detail at pages 15-17 of the specification. Since each frame has the same type of bytes, the receiver can identify and separate the audio bytes from the video bytes. The frame structure allows the users to interleave audio and video bytes in the same frame. This features efficiently uses more of the bandwidth since it is likely each frame will have some information (audio or video) and it is less likely that there will be an empty frame.

The narrowband video codec has a first digital signal processor 304 for handling transmitted video information. The DSP 304 converts analog video signals into digital video signals and compresses the digital video into video bytes. A second digital signal processor 404 handles reception of video digital signals. It decompresses digital video bytes into digital video signals and converts decompressed digital video signals into analog video signals. A third digital signal processor 504 handles both transmission and reception of audio. For transmission it converts analog audio signals into digital audio signals and compresses the digital audio signals into audio bytes. For reception it decompresses the audio bytes into digital audio signals and converts the decompressed digital audio signals into analog audio signals. The invention also provides control and error correction bytes, software and hardware. See Figs. 7a-7d and pp. 15-17.

The invention is a unique combination of hardware and software. It uses separate DSPs to process video input and output and one DSP to process audio. By separating audio from video data, each DSP can operate at optimal efficiency for the type of data (audio or video) that it handles. The invention arranges data into a unique frame structure. Each frame comprises a number of bytes. Each byte includes data of only one type: control, audio, video or error correction. The structure, once chosen, is the same for all frames. It is particularly characterized by one audio byte followed by a set number of video bytes.

In each sequential frame the number of video bytes following the audio byte is the same. As such, the system knows that the next byte after a set number of video bytes will be an audio byte. The system routes the audio bytes to or from the audio DSP 504. An audio algorithm compresses and decompresses the audio bytes. However, the video bytes are handled separately by two DSPs that use algorithms specially tailored for compression or decompression of video data signals. In this way the invention simultaneously uses two or more different algorithms to compress and decompress the audio and video data signals. Regardless of the compression or decompression algorithm, all data bytes have the same number of bits and are treated the same by the rf link.

The invention is claimed in three independent claims and twenty-seven dependent claims. Claim 1 is one independent claim; claims 2-8 depend ultimately from claim 1. Claim 9 is a second independent claim; claims 10-28 depend ultimately from claim 9. Claim 29 is a third independent claim and claim 30 depends from claim 29.

Claim 1 defines the invention in terms of its frame structure for the transmitted and received bytes. That structure is the same for all frames and includes, in sequence: two control bytes, a plurality of sequential data bytes with an audio byte followed by a video bytes, and a plurality of error correction bytes. The audio and video bytes are the same order in each set of data bytes. Claim 2 limits the video bytes to the same number in each frame. Claim 3 further defines the role of the control byte as including information about the number of bytes in each frame. Claim 4 provides for periodically refreshing the compressed video image. Claim 5 provides for variable error correction and claim 6 synchronizes the frames to the radio frequency link. The battery and its voltage are defined in claims 7 and 8.

Dependent claims 9-18 define the detailed frame structure for the bytes. Those claims are designed to cover, *inter alia*, features of the table found on page 17 of the specification. These claims define the individual species of the generic byte arrangement of Claim 1. Certain claims cover the detailed separation of audio bytes by set numbers of video bytes. See, for example, claims 13 and 16.

Independent claim 19 defines the invention in terms of its separate DSPs. There is one DSP for receiving and decompressing video, one for transmitting and compressing video and one that receives, decompresses, transmits and compresses audio.

Dependent claims 21 and 22 define the battery supply and are similar to claims 7 and 8.

Dependent claims 23-28 further define the invention in terms of its radio frequency features. These include its ability to randomize and derandomize data (claims 24, 25) and different modes of clarity (claims 25-28) that let the user select whether to optimize operation for audio or video and for the level of desired video performance.

Independent claim 29 is similar to claim 19 and includes the further feature of multiple compression and decompression algorithms on each DSP. Claim 30 selects a default audio conversion program in accordance with the data rate of the radio frequency link.

6. Issues

The only issue is whether the invention is patentable over the combinations of references of record.

7. Grouping of Claims

Claims 1-18 stand or fall together.

Claims 19-30 stand or fall together.

8. Argument

Claims 1-18 are rejected under 35 USC 103(a) as being unpatentable over Kuzma (US 5389965) in view of Yurt et al. (US 6002720) and Paneth et al. (US 5119375). The rejection alleges that Kuzma shows all the elements of Claim 1 except for the defined byte sequence in a frame and that Yurt shows that sequence.

Claims 19-30 are rejected under 35 USC 103(a) are rejected under 35 U.S.C. 103(a) as being unpatentable over Kuzma in view of Peters of record (5,577,190) and Rostoker et al of record (5,784,572). The rejection finds that Kuzma inherently discloses the three digital signal processors of independent Claims 19 and 29.

Applicant disagrees.

Applicant requests the Board examine two erroneous factual findings made by the Examiner. The first error is a finding that the Yurt reference has the same sequence of bytes as defined in claim 1. The "frames" identified in the rejection are hypothetical constructs made by the Examiner. The hypothetical frames of the rejection are contrary to the express terms of the Yurt reference. Inherent protocol requirements for transmitting data preclude the hypothetical frames from any practical application in a data communication system. The second error is the finding that there are three digital signal processors in the Kuzma reference. That finding is contrary to the express disclosure in Kuzma (there are no DSPs in

Kuzma) and Kuzma has no actual or inherent DSPs. Moreover, the Kuzma reference does not divide the work of the DSPs in the manner called for in the claims.

Claims 1-19 are patentable

Applicant contested the finding that Yurt showed video and audio bytes in the same frame. The Examiner persisted in the rejection. The final office action appears to give no weight to the express language in Yurt that his Fig. 8d showed frames. In that action the Examiner wrote as follows:

> And contrary to the applicant's statement, it is clearly shown in Figure 8d that the bytes within each frame are separated from each other by a different type of data byte. As such it is submitted again that the framing of audio and video data as shown in Figure 8d based on the realignment of audio and video data and user addressing of the date provides the same plurality of video bytes between each sequential audio byte, at least one of the plurality of video bytes between each sequential audio byte, and where each set of data bytes has the same number of video bytes between sequential audio bytes as claimed.

Figure 8d shows items 1, 2 and 3. Each item is made up of frames, such as frames 812, 832 and 822. Each of those frames contains only video or audio bytes. No one frame contains both audio and video bytes.

In order to read the reference on the claims, the rejection ignores the express language of the reference and reads items 1, 2, and 3 as hypothetical frames. Then items 1, 2 and 3 become frames and within each there is a sequence of a video bytes and audio bytes from the actual frames. However, such a construction of the reference is unsupported by and contrary to the reference which clearly identifies elements 812, 832 and 822 as frames. It is also contrary to the inherent frame structure of Yurt.

The hypothetical frame structure of the rejection is incompatible with inherent requirements of a communications protocol. Yurt has no details of its control and error correction bytes, they must exist for each frame and, if they do, then they are part of the respective actual audio and video frames. Control and error correction bytes are inherent because they are an essential part of every communication protocol. Since they are inherent in Yurt, then the Yurt reference must have control and error correction bytes within each frame. With its inherent control and error correction bytes, the hypothetical frame of Yurt does not show or suggest the claimed sequence of bytes.

Those skilled in the art would readily acknowledge that all data communication systems have control and error correction features. In support of this assertion, Applicant attaches a copy of Chapter 7 the reference *Understanding Data Communications*, Griend et al., 1984 (Exhibit A). See page 163 where protocols are discussed.

The Yurt reference describes a system for processing orders for remotely stored video and audio files. It allows a user to request a movie via a communication network rather than renting a physical copy. As such, Yurt provides little or no details about the communication protocol used to transmit its data. Figures 8a-e and several paragraphs spanning columns 19 and 20 provide a high level description with very little detail about the transmission of the data in the network. Yurt says that one may use ISDN, ordinary telephone lines, microwave communications or other communication channel. The details to the communication channel and its protocol are not important to Yurt. However, as explained above, all communication channels have a protocol and that protocol must included error correction and other control bytes. When the control and error correction bytes are added to the hypothetical frames of the rejection, those frames do not have the sequence of audio and video bytes called for the in the claims. Stated another way, the only way the hypothetical frame of Yurt can read on the claims is to ignore the reality of the necessary control and error correction bytes.

<u>Claims 19-30 are patentable</u>

Claims 19-30 are rejected based on the erroneous finding that a codec is inherently a digital signal processor. Claims 19 and 29 call for three DSPs. The art of record relied upon by the rejection is Kuzma. The rejection finds three DSPs in the reference.

The term "codec" may mean a modem that performs A to D and D to A conversions and then codes and decodes a telephone line signal. It is the equivalent of a modem. The term is also used to mean a compression/decompression algorithm, such as discrete cosine transforms, or an integrated circuit that performs a compression/decompression algorithm on a set or pulses. Attached is a copy of a data sheet for a typical codec (Exhibit B). A digital signal processor is an integrated circuit that has incorporated into it an A to D and D to A converter and one or more programs for processing the digitized signals.

A codec chip or circuit has one compression and decompression algorithm. Another algorithm requires another codec for that algorithm. In the invention the DSPs have several compression and decompression algorithms stored as programs. The invention uses on DSP to transmit video and another to receive it. The third DSP handles the transmission and reception of audio. The three DSPs hold one or more algorithms to process the signals. Typical compression and decompression programs are identified in Table A on page 13 of the specification.

The rejection is clearly erroneous because there are no DSPs in the Kuzma reference. Instead it has two conventional codec devices 500, 185 that separately process the audio and video signals and a host processor 160 that is a conventional Ethernet controller chip. Kuzma states in pertinent part that "[a]suitable host processor is the MC68302 which is commercially available from Motorola. In the receiving direction, processor 160 performs the reverse function." The MC68302 is a communication controller chip. It is not digital signal processor nor is it even a general purpose microprocessor. Attached is a copy of a data sheet (Exhibit C) from Motorola's web site that explains the MC68302. That web site has descriptions of DSPs. A typical DSP data sheet is over 100 pages long and is not included in this appeal. However, data sheets are available for inspection at Motorola's product library at http://e-www.motorola.com/webapp/sps/library/prod_lib.jsp.

In the final office action the rejection recognized that the Kuzma reference does not expressly show three DSPs. However, the rejection erroneously found that the reference inherently has three DSPs. See Paper No. 20, pages 7-8. For example, the rejection stated that the reference "must inherently include" a DSP in order to perform the video codec function. That is erroneous because an ordinary codec chip can do that. Attached is a copy of a data sheet for an audio codec that permits voice over the Internet. It is not a DSP.

MPEP Section 2112 places the burden on the Examiner to show that a device has an inherent component. The Examiner failed to make such a prima facie showing. Instead, the word "inherent" is merely applied to the two codec and the controller chip without any support. Even if the Examiner made a prima facie showing of inherency, that showing has been rebutted by Applicant's showing that codecs are not inherent DSPs and the particular host processor of the reference is not a DSP.

As a further showing, Applicant ran a series of key word searches on the USPTO web site. The searches looked for the terms "codec" and "DSP" and "digital signal processor" in patents. The searches show that there are about 1773 patent with the terms codec and DSP, there are 6382 patents with the term "codec", and there are over 23,000 patents with the terms "DSP" or "digital signal processor." While such searches are not conclusive, they provide strong evidence that those skilled in the art recognize that DSPs are different from codecs.

The rejection is also erroneous because Kuzma does not use two DSPs (or two codecs) to separately transmit and receive video and nor does it use one DSP to transmit and receive audio. The invention divides the workload among the DSPs in a manner not shown or suggested by the art of record.

In summary, the rejection is fatally defective because Yurt does not and cannot show the audio/video bit sequence if Claim 1, the Kuzma reference has no inherent DSPs and Kuzma does not use two DSPs for video and one for audio. An order reversing the rejection is respectfully requested.

Respectfully requested,

[illegible]

Thomas R. FitzGerald
Reg. No. 26,730

Law Office of Thomas R. FitzGerald
16 East Main Street, Suite 210
Rochester, NY 14614
Tel: (585) 454-2250
Fax: (585) 454-6364

APPENDIX

1. A narrowband video codec for generating an output stream of control, data, and error correction bits, said narrowband codec comprising:

means for framing the output, control and data bits into a series of sequential frames of bytes for transmission over an rf link of a controlled frequency wherein each frame comprises an identical sequence of bytes;

each frame comprising, in sequence:

two control bytes;

a plurality of sequential sets of data bytes,

each set of data bytes comprising:

a sequence of at least one audio byte and a plurality of video bytes, at least one of said plurality of video bytes between each sequential audio byte, each set of data bytes having its audio and video bytes in the same order as each other set of data bytes; and

a plurality of error correction bytes.

2. The narrowband video codec of claim 1 wherein each set of data bytes has the same number of video bytes between sequential audio bytes.

3. The narrowband video codec of claim 1 wherein the control bytes include data bit signals representative of the number of bytes in the frame.

4. The narrowband codec of claim 1 further comprising means for periodically refreshing the decompressed video image.

5. The narrowband codec of claim 1 further comprising means for controlling of the level of error correction without re-transmitting corrupted data.

6. The narrowband codec of claim 1 further comprising means for synchronizing the frames to the data rate of the rf link.

7. The narrowband codec of claim 1 further comprising a battery power supply.

8. The narrowband codes of claim 7 wherein the power supply voltage is between 18 and 36 volts.

9. The narrowband video codec of claim 1 wherein each frame comprises 200 bytes including two control bytes, 180 data bytes and 18 error correction bytes.

10. The narrowband video codec of claim 1 wherein each frame comprises 150 video bytes and 30 audio bytes.

11. The narrowband video codec of claim 10 wherein sequential audio bytes are separated from each other by five video bytes.

12. The narrowband video codec of claim 9 wherein each frame comprises 165 video bytes and 15 audio bytes.

13. The narrowband video codec of claim 10 wherein sequential audio bytes are separated from each other by eleven video bytes.

14. The narrowband video codec of claim 1 wherein each frame comprises 40 bytes including two control bytes, 18 data bytes and 20 error correction bytes.

15. The narrowband video codec of claim 14 wherein each frame comprises 12 video bytes and 6 audio bytes.

16. The narrowband video codec of claim 15 wherein sequential audio bytes are separated from each other by two video bytes.

17. The narrowband video codec of claim 14 wherein each frame comprises 15 video bytes and 3 audio bytes.

18. The narrowband video codec of claim 15 wherein sequential audio bytes are separated from each other by five video bytes.

19. A narrowband video codec for transmitting and receiving compressed video and audio data signals over a rf link comprising:

a first digital signal processor for converting analog video signals into digital video signals and for compressing the digital video signals into video bytes;

a second digital signal processor for decompressing received digital video bytes into digital video signals and for converting the decompressed digital video signals into analog video signals;

a third digital signal processor for converting analog audio signals into digital audio signals, for compressing the digital audio signals into audio bytes, for decompressing received audio bytes into digital audio signals, and for converting the decompressed digital audio signals into analog audio signals;

means for periodically refreshing the transmitted video signals;

means for running multiple compression and decompression algorithms on all three digital signal processors;

a solid state memory; and

means for emulating a disk access system of a computer using solid state memory components to store file sequences with compression/decompression algorithm data.

20. The narrowband video codec of claim 19 wherein the period for video image refreshing is thirty seconds.

21. The narrowband codec of claim 19 further comprising a battery power supply.

22. The narrowband codec of claim 21 wherein the power supply is between 19 and 36 volts.

23. The narrowband codec of claim 19 further comprising means for sensing the data rate of the rf link and for transmitting and receiving data frames in accordance with the data rate of the rf link.

24. The narrowband codec of claim 19 further comprising means for randomizing data in order to maximize the efficiency of data transmission over the rf link.

25. The narrowband codec of claim 19 further comprising means for de-randomizing data from the rf link without introducing additional bit errors.

26. The narrowband video codec of claim 19 further comprising means for selecting one of a plurality of video resolution and clarity modes.

27. The narrowband codec of claim 26 wherein said video resolution modes include a low resolution mode and a high resolution mode.

28. The narrowband codec of claim 26 wherein said video clarity modes include a low clarity mode, a high clarity mode, and an intermediate clarity mode.

29. A narrowband video codec for transmitting and receiving compressed video and audio data signals over a rf link comprising:

a first digital signal processor for converting analog video signals into digital video signals and for compressing the digital video signals into video bytes;

a second digital signal processor for decompressing received digital video bytes into digital video signals and for converting the decompressed digital video signals into analog video signals;

a third digital signal processor for converting analog audio signals into digital audio signals, for compressing the digital audio signals into audio bytes, for decompressing received audio bytes into digital audio signals, and for converting the decompressed digital audio signals into analog audio signals;

means for periodically refreshing the transmitted video signals;

means for running multiple compression and decompression algorithms on all three digital signal processors;

a solid state memory;

means for emulating a disk access system of a computer using solid state memory components to store file sequences with compression/decompression algorithm data; and

a memory for storing a program connected to at least the third digital signal processor, said memory comprising at least two audio conversion programs for converting audio at first and second respective rates.

30. The narrowband codec of claim 29 further comprising means for automatically selecting one of said audio conversion programs in accordance with the data rate of the rf link.

A

Protocols and Error Control

ABOUT THIS CHAPTER

This chapter is about the rules that data communications systems use when communicating with each other, how they discover that an error has been made in transmission, and some methods that are used in correcting those errors. Since there are many data transmission schemes in use (for example, asynchronous and synchronous, half- and full-duplex), many sets of rules, called protocols, and many error detection and correction schemes have been devised. The most common of these techniques are explained in this chapter.

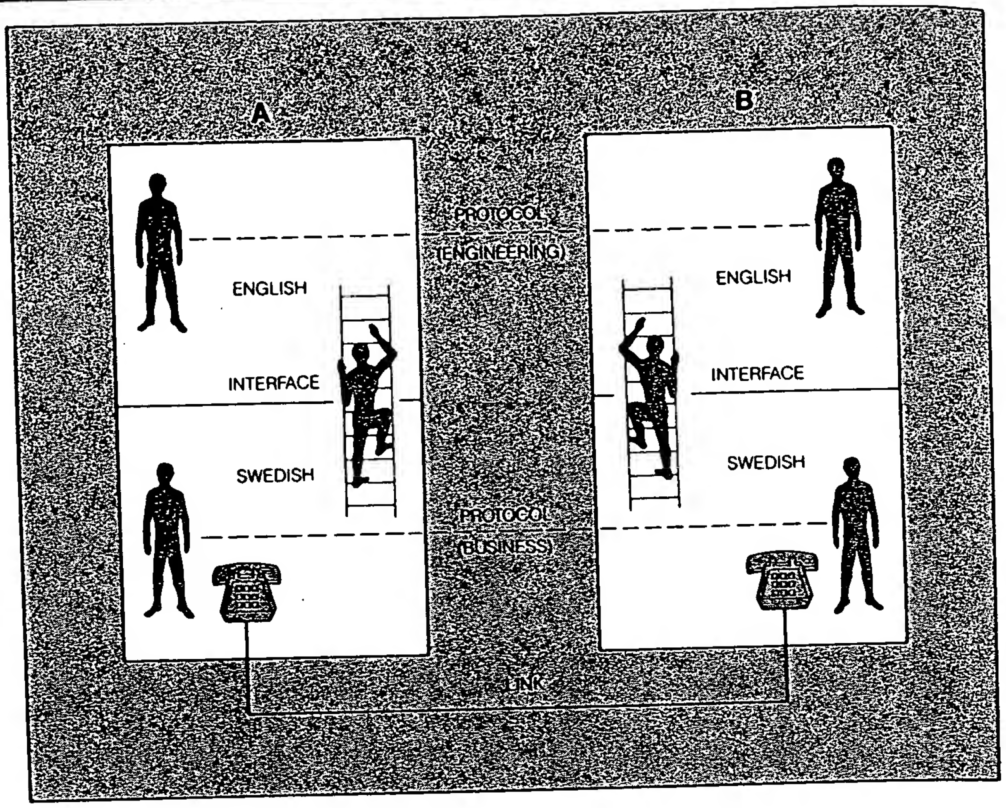
PROTOCOLS AND INTERFACES

A protocol is a set of rules defining the interactions between two machines or processes that are alike or that have similar functions. An interface is a set of rules controlling the interaction of two *different* machines or processes.

Protocols are agreements between people or processes (usually nowadays the processes are computer programs) about which may do what to whom, and when. A protocol is different from an interface. An interface is a set of rules, often embodied in pieces of hardware, controlling the interaction of two different machines or processes, such as a computer and a modem. A protocol, on the other hand, is a set of rules defining the interactions between two machines or processes that are alike or that have similar functions. The difference is illustrated in *Figure 7-1.* House A and House B look alike, but are occupied by a curious set of people. The British engineer on the top floor of house A needs to communicate with another engineer, also British, on the top floor of B. The only telephone, however, is guarded by a Swedish businessman on the bottom floor of each house. The engineer speaks no Swedish, and the Swede no English.

The two engineers, like most people who talk on the telephone, have a routine, worked out over long years, for communicating. The one answering says "Hullo." The one calling says "Hullo, this is Reggie." Then the first says "Oh, hullo, Reggie. How are you, old boy?" This is called a preamble. They then begin discussing Wimbledon, the weather, and even some business. At the end of most sentences, the one being questioned makes a reply, even if it is only a sound rather than a word or sentence. If one party does not hear some sort of reply from the other at fairly short intervals, he may say, "I say, are you still there?" If he still receives no reply, he may terminate the conversation and begin the whole process over. If one party does not understand the other, he will say, "What?", and the second party will repeat the last sentence. When the conversation is over, the person terminating the call will say, "Well, goodbye for now." The called party will reply, "Goodbye.", and both will hang up. The

Figure 7-1.
Protocols and Interfaces



Swedish businessman has a similar technique for conversing with his counterpart. All of these conventional conversational actions make up a protocol. As we shall see in the remainder of this chapter, protocols developed for communicating between processes in computers have all of the same mechanisms as those used between humans, for exactly the same reasons.

Protocols function for machine communication much like the formats and rules for human conversations and are used for the same reasons.

Even though our two engineers are able to communicate reliably with each other because they have a communications protocol, they still have the problem of no direct access to the communications channel, which is jealously guarded by the large Swedish businessmen. The solution is an interface. The engineers engage a person who stands midway up the stairs and who speaks both engineering English and business Swedish, and whose task it is to relay the conversation from the engineer (speaking in English) to the businessman (listening in Swedish), who relays it over the telephone to his Swedish counterpart in house B, who relays it in Swedish to another person halfway up the stairs, who finally conveys the messages in English to the called party. This seems like a clumsy and inefficient procedure and in many respects it is, but it is the same procedure used in millions of data communications systems to allow an applications program, communicating in one set of symbols (perhaps ASCII) and at one rate of symbol production, to communicate through a teleprocessing monitor using another set of symbols (perhaps EBCDIC) at a vastly different rate, to a line interface program dealing with yet a third set of symbols (bits). The story of standard interfaces is told in chapter four. The story in this chapter is about protocols.

Elements of a Protocol

There are three main elements to any protocol: 1) a character set, 2) a set of rules for message timing and sequence, and 3) error determination and correction procedures.

The basic elements of a communications protocol are a set of symbols, called a character set, a set of rules for the sequence and timing of messages constructed from the character set, and procedures for determining when an error has occurred in the transmission and how to correct the error. The character set will consist of a subset which is meaningful to people (usually called printing characters), and another subset which conveys control information (usually called control characters). There is a correspondence between each character and a group of symbols on the transmission channel. For instance the printing character A might correspond to the binary code 1000001. Several standard codes with equivalent sets of ones and zeros (bits), such as the ASCII discussed in a previous chapter, have been defined over the years. The set of rules to be followed by the sender and receiver gives the meaning, permissible sequences, and time relationships of the control characters and messages formed from the symbols. The error detection and correction procedure allows for the detection of and orderly recovery from errors caused by factors outside the control of the terminal at either end.

TELETYPEWRITER AND XMODEM PROTOCOLS

TTY protocols are the simplest ones in general use. The 58 characters in the character set of one TTY protocol were encoded by a five-bit binary code.

The simplest protocols in general use are the ones associated with the transmission of start-stop or asynchronous data between teleprinter machines. For a long period after the development of the teleprinter, the machines were electromechanically controlled as we discussed in a previous chapter, and protocols had to be appropriate to a very simple mechanical controller in each machine. One of the first teletypewriter (TTY) protocols used a character set containing 58 symbols: 50 printing characters, a space, and seven control characters as shown in *Figure 7-2*. These symbols were encoded into a 5-bit binary code as shown (the octal representation is shown only for readers familiar with octal). The printing characters are the upper case letters, the numbers, and 14 punctuation marks. Three control characters allow for carriage return (CR), line feed (LF), and ringing the terminal bell (BELL) to announce a message. Also provided are a NULL (or blank) character (which is not the same as a space) and a WRU character. WRU stands for "Who Are You", and causes a mechanical device in the teleprinter to send a character sequence identifying that particular terminal. The other two control characters are the FIGS character, which causes the machine to print numbers and punctuation, and the LTRS character, which causes the machine to print the upper case alphabet. As mentioned in a previous chapter, this character set and associated binary code is commonly referred to as the Baudot code, after Emile Baudot who invented the first code with symbols of a fixed length.

Figure 7-2.
CCITT Alphabet #2

BINARY	OCTAL	LTRS	FIGS
0 0 0 0 0	00	BLANK	BLANK
0 0 0 0 1	01	E	3
0 0 0 1 0	02	LF	LF
0 0 0 1 1	03	A	-
0 0 1 0 0	04	SPACE	SPACE
0 0 1 0 1	05	S	/
0 0 1 1 0	06	I	8
0 0 1 1 1	07	U	7
0 1 0 0 0	10	CR	CR
0 1 0 0 1	11	D	WRU
0 1 0 1 0	12	R	4
0 1 0 1 1	13	J	BELL
0 1 1 0 0	14	N	,
0 1 1 0 1	15	F	
0 1 1 1 0	16	C	:
0 1 1 1 1	17	K	(
1 0 0 0 0	20	T	5
1 0 0 0 1	21	Z	+
1 0 0 1 0	22	L	)
1 0 0 1 1	23	W	2
1 0 1 0 0	24	H	
1 0 1 0 1	25	Y	6
1 0 1 1 0	26	P	0
1 0 1 1 1	27	Q	1
1 1 0 0 0	30	O	9
1 1 0 0 1	31	B	?
1 1 0 1 0	32	G	
1 1 0 1 1	33	FIGS	FIGS
1 1 1 0 0	34	M	.
1 1 1 0 1	35	X	/
1 1 1 1 0	36	V	=
1 1 1 1 1	37	LTRS	LTRS

The protocol for operation of a message system using the Baudot code goes something like this (the procedure varies somewhat according to the operator of the system):

1. A communications channel is connected between the two machines. This may be a temporary, dialed connection, or a permanent private line.
2. The sending machine sends a WRU, to verify that the receiving machine is the proper one for the message to be sent.
3. The sending machine sends a "Here is ...", which is normally the same sequence of characters which is sent when it receives a WRU, to identify the sender.

4. The sending machine sends a preamble or message header which identifies the name and address of the intended message recipient, the date and time of entry of the message into the system, the date and time of transmission, and the message sequence number assigned by the sender. In some systems, the sender will use two sequence numbers; one to count the number of messages sent by the sender that day, and the second to count the number of messages sent to the receiver's terminal that day.
5. At the end of the message, the sending machine sends another "Here is . . ." and another WRU. The second WRU is to determine if the receiving machine is still connected to the line. The operator of the sending machine will, if the message is being sent manually, sometimes send several BELL characters, to alert a person at the receiving end that a message is ready for delivery.

Because the simple TTY protocol does not check for errors in the message, even low error rates will cause the received message to be gibberish and the sender will not know it.

This protocol has developed over many years of sending messages manually between individual terminals operated by people. You will note that it has all of the elements described in the opening paragraphs: a previously defined set of symbols and corresponding codes suitable for electronic transmission, a preamble, a message, and even a rudimentary error detection procedure (the invocation of the WRU at the beginning and end of the session). The same sort of protocol is used for teleprinter message systems even when one end of the system is a computer, as are all of the current public teleprinter networks.

The simple teletypewriter protocol described above does not work very well in the presence of even fairly low error rates on the path between the machines. Individual characters can be mutilated in such a way as to cause the receiving machine to begin printing gibberish, and the transmitter will never know. The WRU check at the end of a message assures only that the receiving machine received the WRU correctly and that the circuit is still good in the backwards direction, but says nothing about whether the message got through. To provide better assurance of correct transmission, two more techniques are sometimes used for simple protocols: character parity and echoplex.

Parity

Parity is the technique where a bit is added to every symbol for the purpose of error detection. It can either be even or odd parity.

The Communicator's Creed: "Now abideth faith, hope, and parity, but the greatest of these is parity."

The addition by the transmitter of another bit to the bits that encode a symbol for the purpose of error detection is called parity. The bit is always transmitted, and is usually set to the value that will cause the coded symbol to have an even number of one bits; thus, the scheme is called even parity. The parity bit is recomputed by the receiver. If the newly computed value gives the correct parity, all is well. If not, some indication is given to the receiving terminal, usually by substituting a special error character for the one received.

The parity scheme will detect single-bit errors in the transmitted symbols, but not multiple-bit errors. Error correction can occur by the receiver sending a message back which requests retransmission.

Echoplex

Echoplex is an error detection technique in which each character of a message is retransmitted by the receiver back to the originator as it is received.

Echoplex is perhaps not properly classed as an element of protocols, but we will discuss it here anyway. It is the technique of sending back (or echoing) each character by the receiver as the characters are received. The sender can then see by the copy printed locally whether the characters are making the round trip without being mutilated. When an echoed character is received in error by the original sender, it is not possible to tell whether the data were received correctly at the destination and scrambled on the return path, or were erroneous when received at the original destination. But at least some error indication is immediately available to the sender.

Checksums

In addition to character (or vertical) parity where a parity bit is added to each character, a block parity character may be added at the end of each message block.

In the discussion on parity above, the bit required to make the number of 1-bits in an individual character was added onto the end of the character and sent along with it. This scheme is sometimes called vertical redundancy checking (VRC), or vertical parity, because if one holds a punched paper tape with its length in the horizontal, each character will appear as a vertical column of holes across the tape. It is possible, and is common in some systems, to include a horizontal check character, which performs the parity function for each row of holes in the tape.

Figure 7-3 shows a set of characters represented as holes punched in a tape, with the vertical or character parity bit at the top of each character, and the horizontal or block parity character at the right. The block parity character is usually called a checksum, because it is formed by performing a binary addition without carry of each successive character. It also is sometimes

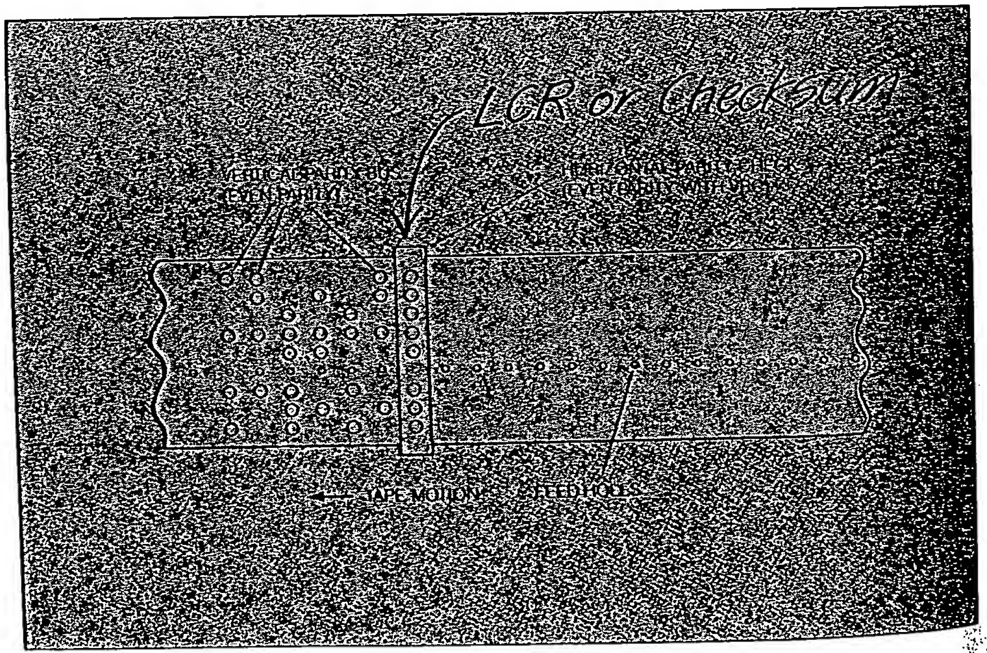

LCR or Checksum
VERTICAL PARITY BITS (EVEN PARITY)
HORIZONTAL PARITY CHECK (EVEN PARITY WITH VRC)
TAPE MOTION
FEED HOLE

Figure 7-3. Vertical and Horizontal Parity

referred to as a longitudinal redundancy check (LRC) character. It is sent as an extra character at the end of each message block. A system which uses both vertical parity and a checksum can usually detect all single-bit errors in a single character, and some multiple-bit errors within a single character.

XMODEM Protocol

XMODEM protocol has an error checking technique that can be used between microcomputers. Messages are sent in blocks of 128 characters surrounded by control characters.

The XMODEM protocol was devised by Ward Christensen as a simple error checking scheme suitable for use between microcomputers. It requires that one terminal or computer be set up by an operator or computer program as the sender, and the other be set up as the receiver. Once the protocol is started, the transmitter waits for the receiver to send a Negative Acknowledge (NAK) character. The receiver meanwhile is set to sending NAKs every 10 seconds. When the transmitter detects the first NAK, it begins sending messages as blocks of 128 data characters, surrounded by some protocol control characters. The beginning of each block is signaled by a Start Of Header (SOH) character. This is followed by a block number character in ASCII, followed by the same block number with each bit inverted. A 128-character piece of the file is sent, followed by a checksum which is the sum of all of the 128 bytes in the message.

The receiver checks each part of the received block as follows:

1. Was the first character an SOH?
2. Was the block number exactly one more than the last previous block received?
3. Were exactly 128 characters of data received?
4. Was the locally computed checksum identical with the last character received in the block?

If the receiver is satisfied, it sends an Acknowledge (ACK) back to the transmitter, and the transmitter sends the next block. If not, a NAK is sent, and the transmitter resends the block found in error. This process is continued, block by block, until the entire message or file is sent and verified. At the end of the data, the transmitter sends an End Of Text character. The receiver replies with an ACK and the session is terminated.

Even though the XMODEM protocol is very simple and easy to operate, it requires a computer at each end, and every file to be transferred must be set up manually.

There are several points to be made about the XMODEM protocol. First, it is very simple and easy to implement with a small computer, but it does require a computer at each end. Second, it requires manual setup for each file to be transferred. Third, the error detection technique (ordinary sum of the data characters) is unsophisticated and unable to detect reliably the most common type of transmission error, which is a noise burst which may last on the order of 10 milliseconds (the duration of about 12 bits at 1200 bps). Fourth, it is a half-duplex protocol; that is, information is sent, then the sender waits for a reply before sending the next message. Since operation of the XMODEM protocol generally assumes a full-duplex line, it is inefficient in use of the transmission facility.

CONVOLUTIONAL CODING — CYCLIC REDUNDANCY CHECKS

Convolutional coding attaches a BCC to the end of each message block. The receiver recomputes BCC and compares it to the one transmitted to determine if the message was correctly received.

Several schemes have been devised to detect errors in binary communications systems using feedback or convolutional coding. These methods all append information computed at the transmitter to the end of each message transmitted to enable the receiver to determine if a transmission error has occurred. The added information is mathematically related to the messages and is therefore redundant. The receiver recomputes the value and compares the recomputed number with the number received. If the two are the same, all is well. If not, the receiver notifies the transmitter that an error has occurred, and the message is re-sent. These methods go by the name of Cyclic Redundancy Checking (CRC) and the values appended to the messages are called CRCs or Block Check Characters (BCCs). A CRC is calculated by dividing the entire numeric binary value of the block of data by a constant, called a generator polynomial. The quotient is discarded, and the remainder is appended to the block and transmitted along with the data.

The check character is determined by dividing the total binary value of the entire block by a constant called a generator polynominal.

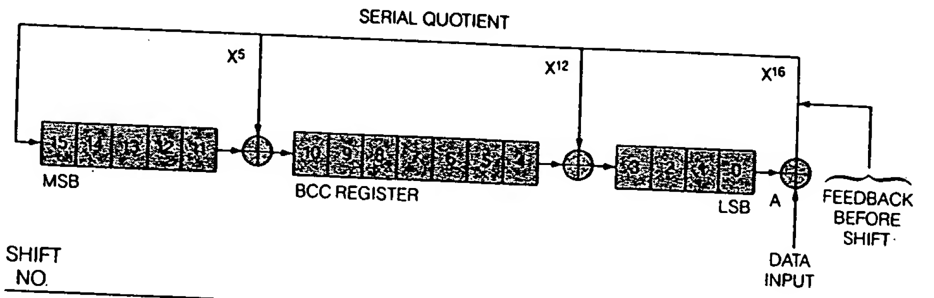
CRCs are usually computed using multiple section feedback shift registers with eXclusive-OR (XOR) logic elements between each section and at the end. A typical arrangement is shown in *Figure 7-4*. This register implements the CCITT/ISO High-level Data Link Control (HDLC) CRC, which is called CCITT-CRC. The circles with a + in the middle represent XOR logic elements. For B = 0 or B = 1, the two rules for XOR are:

1. B XOR B = 0
2. B XOR 0 = B

The shift register circuit is initialized to all ones at the beginning of CRC calculation for a message. As each bit of the transmitted characters is applied to the transmission facility, it is also applied at point A of *Figure 7-4*, and then the entire register is shifted right one bit. As the bits are transmitted and shifted, each one-bit that appears at A also affects the state of the other XOR elements, and that effect is propagated throughout the register for several bit times after the bit initially appears. Thus any bit continues to have an effect on the transmitted data for a considerable time after that bit is sent. When the last data bit has been sent, the bits in the CRC shift register are complemented and transmitted.

At the receiver, the identical process is performed, and when the end of the message including the CRC is detected, the CRC is tested for the unique value 0001110100001111. If this value is found, all is well and the CRC register is reset to all ones for the next message. If the special value is not found, the program is notified that a transmission error has occurred, and a negative acknowledgement is returned to the sender.

Figure 7-4.
Shift Register Implementation of CCITT-CRC
(Source: John E. MacNamara, Technical Aspects of Data Communication, © 1977, Digital Press, Maynard, Mass.)



SHIFT NO.					
Start	0 0 0 0 0	0 0 0 0 0 0 0	0 0 0 0		1
1	1 0 0 0 0	1 0 0 0 0 0 0	1 0 0 0	1	0
2	0 1 0 0 0	0 1 0 0 0 0 0	0 1 0 0	0 LSB	0
3	0 0 1 0 0	0 0 1 0 0 0 0	0 0 1 0	0	0
4	0 0 0 1 0	0 0 0 1 0 0 0	0 0 0 1	0	1
5	1 0 0 0 1	1 0 0 0 1 0 0	1 0 0 0	0	0
6	0 1 0 0 0	1 1 0 0 0 1 0	0 1 0 0	0	0
7	0 0 1 0 0	0 1 1 0 0 0 1	0 0 1 0	0	0
8	0 0 0 1 0	0 0 1 1 0 0 0	1 0 0 1	0	1
9	1 0 0 0 1	1 0 0 1 1 0 0	1 1 0 0	0	0
10	0 1 0 0 0	1 1 0 0 1 1 0	0 1 1 0	0	0
11	0 0 1 0 0	0 1 1 0 0 1 1	0 0 1 1	0	1
12	1 0 0 1 0	1 0 1 1 0 0 1	0 0 0 1	0	1
13	1 1 0 0 1	1 1 0 1 1 0 0	0 0 0 0	0	0
14	0 1 1 0 0	1 1 1 0 1 1 0	0 0 0 0	0	0
15	0 0 1 1 0	0 1 1 1 0 1 1	0 0 0 0	0	0
16	0 0 0 1 1	0 0 1 1 1 0 1	1 0 0 0	0 MSB	0

Arrows indicate Exclusive-Or of data bit and LSB of BCC register prior to shift

Block Check Character (BCC)

16 Bit Data Word

NOTES

□ = BCC Register Stage
⊕ = Exclusive-OR
CRC – CCITT Polynomial = $X^{16} + X^{12} + X^5 + 1$

The CRC error detection process has a very high rate of success in accurately detecting error bursts of up to 16 bits because of the history held in the shift registers.

The CRC process has the advantage that the current state of the shift register is the result of considerable past history. It is therefore unlikely that a burst of errors, such as normally occurs in serial data transmission, will produce a calculated CRC at the receiver that is the same as that which was originally sent. In fact, the 16-bit CRC calculation procedures, CRC-16 and CRC-CCITT, will detect all error bursts of 16 bits or less in length. They will also detect over 99.9% of error bursts of greater than 16 bits. Another advantage of CRC is that it does not require the addition of another bit per character sent as do the VRC and LRC schemes. It does, however, require the sending of one or two (two for CRC-16 and CRC-CCITT) extra characters at the end of each transmitted block.

CRC algorithms are usually implemented in hardware, and integrated circuits have been developed to do the entire process for more than one method (e.g., CRC-12, CRC-16, and CRC-CCITT). It is also possible to do the calculations in software, with table lookup techniques providing reasonable performance.

HALF-DUPLEX PROTOCOLS

Links

Data links, either point-to-point or multipoint, are the data communications equipment and circuits that allow two or more terminals to communicate.

The basic notion in link protocols is that of the data link. A data link is an arrangement of modems or other interface equipment and communications circuits connecting two or more terminals that wish to communicate. Probably the most widely used link protocol is comprised of the Binary Synchronous Communications procedures defined by IBM[1] (usually abbreviated as BiSync or BSC). These procedures allow for operation on a data link in one direction at one time. BiSync can be operated on full-duplex circuits, but information still flows in only one direction at a time, so that in many cases the advantage of the full-duplex circuit is minimized.

For the purpose of discussion of protocols, the physical form of the link is important in that the procedures for connecting and disconnecting the link may be different depending on whether the link is permanently connected or dial-up, and also depending on the delay between when a data bit is sent and when it is received. This latter factor is of particular concern on satellite links due to the large difference in round-trip propagation delay between satellite links and terrestrial links. Regardless of the physical form of the link, however, the data are sent over it as a serial stream of symbols encoded as bits, and the control procedures between the ends of the link are effected using the transmission and recognition of line-control characters or control codes.

Point-to-Point Links

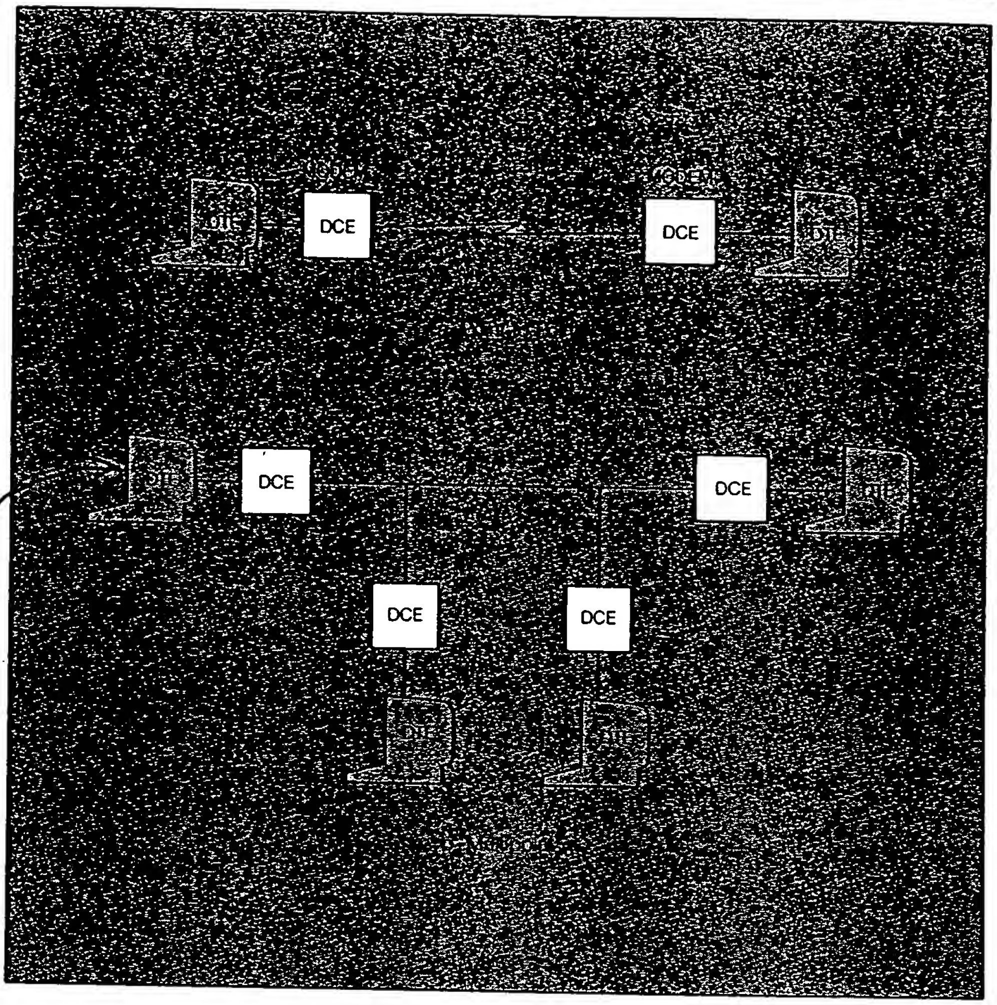
A point-to-point link is one that connects only two stations at one time as shown in *Figure 7-5a*. Point-to-point links may be established on dedicated or dial-up circuits, and they may be half-duplex or full-duplex.

Multipoint Links

Multipoint links connect more that two stations at one time as shown in *Figure 7-5b*. Clearly some control procedure must be in place to designate which stations may use the link at any one time. For this purpose, one station on the multidrop line is designated as the control station, and all other stations are designated as tributaries. The control station is the traffic director, designating which stations are to use the link by a polling and selection process. At any instant in time transmission on a multipoint link will be between only two stations, with all others stations on the link in a passive receive mode.

[1]IBM is a trademark of International Business Machines Corporation.

Figure 7-5.
Point-to-Point and
Multipoint Links



One station is the control station; all others are tributaries.

IBM's BiSync protocol is one of the most widely used link protocols. It can be used with three character sets: Six-Bit Transcode, Extended Binary Coded Decimal Interchange Code, and American Standard Code for Information Interchange.

Transmission Codes — Character Sets

BiSync is defined by IBM to accommodate three character sets and their associated binary codes. Each set consists of a set of graphics (letters, numbers, and punctuation), a set of terminal control and format control codes (BELL, Form Feed, WRU, etc), and a set of data link control codes (Start of Text, End of Transmission, etc). The three sets are the: 1) Six-Bit Transcode (SBT), 2) Extended Binary Coded Decimal Interchange Code (EBCDIC), and 3) United States of America Standard Code for Information Interchange (USASCII, or more commonly, just ASCII).

The codes differ in number of bits per symbol encoded (6 for SBT, 7 for ASCII, and 8 for EBCDIC), and different numbers of characters in the character set (64 for SBT, 128 for ASCII, and 144 for EBCDIC). There are significant differences between the sets in such properties as the sorting order between letters and numbers.

Link Control Codes

Various link control codes are used in BiSync to gain control of the data link and ensure that the proper actions occur.

Link control is effected by the proper recognition of control characters, and the taking of appropriate actions. The control codes used in BiSync are defined as follows:

Synchronous Idle (SYN) — Establishes and maintains synchronization between DCEs, and is used as a filler between data blocks.

Start of Header (SOH) — Identifies the beginning of a block of heading information. Headers are control information (such as addresses, priority, and message numbers) used by the system to process the text part of the message.

Start of Text (STX) — Identifies the end of a header and the beginning of a block of text. Text is the part of the message from an applications program which is destined for another applications program and which must pass through the communications system unchanged.

End of Transmission Block (ETB) — Identifies the end of a block started with an SOH or STX. A BCC is sent immediately following an ETB. Receipt of an ETB requires a status reply (ACK0, ACK1, NAK, WACK or RVI).

End of Text (ETX) — Terminates a block of data started by an STX or SOH which was transmitted as an entity. A BCC is sent immediately following an ETX. ETX also requires a status reply.

End of Transmission (EOT) — Indicates the end of message transmission by this station; the message may have contained one or more blocks. Receipt of an EOT causes all receiving stations to reset. The EOT is also used as a response to a poll when the polled station has nothing to send, and as an abort signal when the sender cannot continue transmission.

Enquiry (ENQ) — Requests a repeat transmission of a response to a message block if the original response is garbled or is not received. ENQ may also indicate the end of a polling or selection sequence, and is used to bid for the line when the line is a point-to-point connection.

Affirmative Acknowledgement (ACK0 or ACK1) — Indicates correct receipt of the last previous block, and that the receiver is ready to accept the next block. ACK0 and ACK1 are used alternately, and receipt of the wrong one is an indication of an error in the protocol. ACK0 is the correct response to a station selection (on a multipoint circuit) or a line bid (on a point-to-point circuit).

Wait-Before-Transmit Affirmative Acknowledgement (WACK) — Indicates a temporary not-ready-to-receive condition to the sending station and affirmative acknowledgement of the last previously received data block. The normal response to WACK by the sending station is ENQ, but EOT and DLE EOT are also valid. If ENQ is received after sending WACK, the sending station continues to send WACK until it is ready to continue sending data.

Negative Acknowledgement (NAK) — Indicates that the last previously received block was received in error, and that the receiver is ready to receive another block. NAK is also used as a station-not-ready reply to a station selection or line bid.

Data Link Escape (DLE) — Indicates to the receiver that the character following the DLE is to be interpreted as a control character.

Reverse Interrupt (RVI) — Indicates a request by a receiving station that the current transmission be terminated so that a high-priority message can be sent (such as a shutdown notice). RVI is also an affirmative acknowledgement to the last previous block received. In a multipoint circuit, RVI sent from the control station means the control station wishes to select a different station. When a sending station receives an RVI, it responds by sending all data that would prevent it from becoming a receiving station.

Temporary Text Delay (TTD) — Indicates that the sending station is not ready to send immediately, but wishes to keep the line. The receiving terminal reply to TTD is NAK. TTD normally is sent after two seconds if the sender cannot transmit the next block; this holds off the normal three-second abort timer at the receiving terminal. The response to TTD is NAK, and TTD may be repeated several times. TTD also is used as a forward abort, by sending EOT in response to the NAK reply.

Switched Line Disconnect (DLE EOT) — Indicates to a receiver that the transmitter is going to hang up on a switched line connection.

Code Sequences

Some control characters require a two-character sequence of standard characters.

Reference to the ASCII and EBCDIC code charts will reveal no code assigned to several of the control characters mentioned above. Such control codes are represented by two-character sequences of the characters that are defined in the charts. *Table 7-1* gives the correspondence between these control characters and sequences of standard characters.

Table 7-1. Character Sequences for BiSync Control Characters

BiSync Data Link Character	Character Sequences for Various Code Sets		
	ASCII	EBCDIC	SBT
ACK0	DLE 0	DLE 70	DLE -
ACK1	DLE 1	DLE /	DLE T
WACK	DLE ;	DLE .	DLE W
RVI	DLE	DLE @	DLE 2
TTD	STX ENQ	STX ENQ	STX ENQ

Polling and Selection

In a BiSync link, the control station determines which tributary station is active on the link, as well as the direction of transmission.

The active participants on a BiSync link are managed by a control station which issues either a Poll or a Select message addressed to the desired tributary. The Poll is an invitation-to-send from the control to the tributary. This allows the tributary to send any messages desired to the control station. The Select is a request-to-receive notice from the control station, telling the tributary that the control station has something to send to it. The control station thus controls which station has the link, and the direction of transmission.

Stations on the data link are assigned unique addresses which may consist of from one to seven characters. The first character defines the station, and succeeding characters define some part of the station, such as a printer, as required. Some BiSync implementations repeat the station address for reliability purposes.

Message Blocks

Messages consist of one or more blocks of text, called the body of the message, surrounded by synchronization, header and error control characters. The beginning of each block is identified by the STX control character, and all blocks except the last in a message are ended either by an ETB or an ITB character. The last block of the message is ended by an ETX character.

Error Checking

Three types of error detection modes are used by BiSync: An odd parity check of each character including VRC/LRC, a BCC comparison using CRC-12, and a BCC comparison using CRC-16.

BiSync implements three types of error detection: VRC/LRC, CRC-12, and CRC-16. *Table 7-2* shows under what conditions each is used. Transparency mode is described later in this section. The VRC is an odd-parity check performed on each character transmitted, including the LRC character at the end of the block. Each bit in the LRC character provides odd parity for the corresponding bit position of all characters sent in the block. In BiSync, this character is called the Block Check Character (BCC) and is sent as the next character following an ETB, ITB, or ETX character. The BCC sent with the data is compared at the receiver with one accumulated by the receiver, and if the two are equal, all is well. The BCC calculation is restarted by the first STX or SOH character received after the direction of transmission is reversed (called a line turnaround). All characters except SYN characters received from that point until the next line turnaround are included in the calculation. If the message is sent in blocks with no line turnaround, each block will be followed by an ITB, then the BCC. The BCC calculation is then restarted with the next STX or SOH character received.

Table 7-2.
Type of Redundancy Check

Code Set	No Transparency	Transparency Installed and Operating	Transparency Installed but Not Operating
EBCDIC	CRC-16	CRC-16	CRC-16
ASCII	VRC/LRC	CRC-16	VRC/CRC-16
SBT	CRC-12	CRC-12	CRC-12

The Cyclic Redundancy Check codes are used for error checking in the same fashion as the LRC code described above. If the transmission code set is SBT, the CRC-12 method is used, since each transmitted character is only six bits. For EBCDIC, the CRC-16 scheme is always used. For the ASCII code set, IBM has specified that the VRC/LRC scheme be used in the non-transparency (standard) mode, and CRC-16 be used in transparency mode. Transparency mode is described in detail below.

Message Formats

Information is carried in BiSync as messages, and each message may have several parts: a synchronization sequence, a header, some text, and a block check sequence. Each part is identified by one or more control characters. In the case of messages used only for control, some parts such as the header, the text, or the BCC may be missing.

Synchronization

Establishing the synchronization required between sender and receiver requires a three step sequence: Bit synchronization, character synchronization, and message synchronization.

Unless the transmitter and receiver agree on the exact (to the bit) starting point of a message, they cannot communicate. Achieving this synchronization requires three steps:

1. The modems or other data communications devices at both ends of the circuit must acquire bit synchronization. The methods for doing this are discussed in Chapter 5.
2. The link interface equipment in the DTE must acquire character synchronization. This is done by searching the bit stream for a specific pattern of bits called a Synchronization Character (SYN). In ASCII, the bit pattern for SYN is 0010110. To help insure against recognition of a false SYN, most systems including BiSync require transmission and detection of two successive SYNs before the next step in the synchronization process is taken. The two SYN characters taken as a control set are called a Sync Sequence, and are shown in most diagrams as a ∅. In order to insure that the first and last characters of a transmission are correctly sent, some BiSync stations add a PAD character before the first SYN and after every BCC, EOT, and ACK. This strategem was devised primarily to overcome hardware limitations in some early hard-wired BiSync terminals.

3. The program operating the protocol must acquire message synchronization. In other words, the program must be able to find the beginning of each message or control character sequence. This is achieved by a program search of the characters received after the sync sequence for a control character which is defined to begin a block of data or control sequence. Such characters are SOH, STX, ACK0/1, NAK, WACK, RVI, and EOT.

Heading

The character SOH begins the heading used in identifying the message type, numbering, priority and routing.

A heading is a sequence of characters beginning with the character SOH that is used for message type identification, message numbering, priority specification, and routing. Receipt of the SOH initiates accumulation of the BCC (but the SOH is not included in the BCC). The heading may be of fixed or variable length, and is ended by an STX.

Text

The text part of the message contains the information to be sent between application programs. The text begins with an STX an ends with an ETX. Text may be broken up into blocks for better error control. Each block begins with an STX and ends with an ETB, except for the last block which ends with an ETX. The normal end of a transmission is signaled by sending an EOT following the ETX. Control characters are not allowed within the body of a text block. If a control character is detected, the receiving station terminates reception and looks for a BCC as the next two characters.

Timeouts

Four Bisync time-outs functions prevent indefinite delays due to data errors or missing line turnaround signals.

Timeouts must be provided by the communications program or terminal to prevent indefinite delays caused by data errors or missing line turnaround signals. Four functions are specified in BiSync for timeouts: transmit, receive, disconnect, and continue.

Transmit timeout is normally set for one second, and defines the rate of insertion of synchronous idle (SYN SYN or DLE SYN) sequences in the data to help maintain bit and character synchronization.

Receive timeout is normally set for three seconds, and does several things:

1. It limits the time a transmitting station will wait for a reply.
2. It signals a receiving station to check the line for synchronous idle characters, which indicate that transmission is continuing. The receive timeout is reset each time a sync sequence is received.
3. It sets a limit on the time a tributary station in a multipoint circuit may remain in control mode. The timer is reset each time an end signal such as an ENQ or ACK is received as long as the station stays in control mode.

Disconnect timeout causes a station on a switched network to disconnect from the circuit after 20 seconds of inactivity.

Continue timeout causes a transmitting station sending a TTD to send another if it is still unable to send text. A receiving station must transmit a WACK if it becomes temporarily unable to receive within a two-second interval.

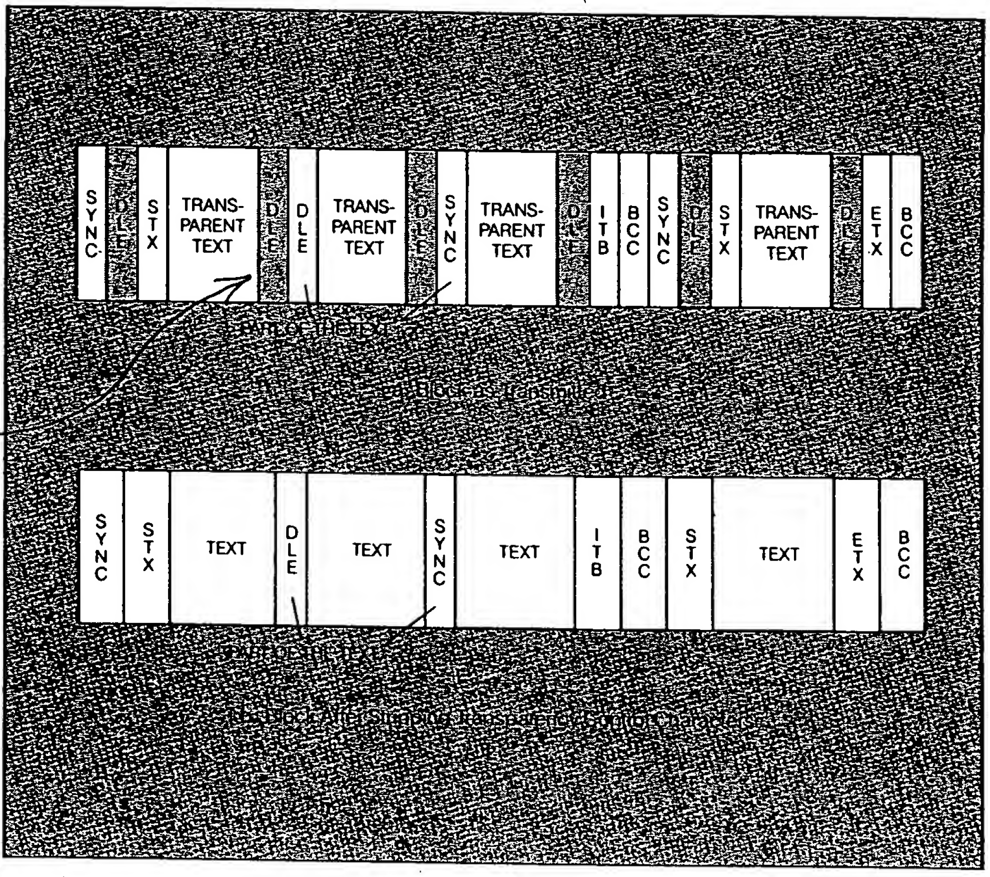
Transparent-Text Mode

Data communications often requires the sending of information that is in binary instead of in a character code. A transparent-text mode is used for this purpose.

It is frequently necessary in communicating between machines to send data which do not represent characters, but instead represent some purely arbitrary quantity or object. An example is the binary representation of a computer program. In such a case, it is likely that some of the data will have the same bit pattern as a BiSync control character. Transparent-text mode, sometimes called transparency, allows such data to be sent without being misinterpreted by the communications program. The basic technique in transparency is to precede each true control character with the DLE character as shown in *Figure 7-6a.* If a DLE bit pattern appears in the text of the message, it also is preceded by a DLE. Thus, a bit pattern is interpreted as a control character only if preceded by a DLE. The resulting message after processing by the link interface program to remove the DLEs is shown in *Figure 7-6b.* Note that any DLE DLE sequence in the text has the first DLE suppressed and the second sent along as part of the data.

Figure 7-6. BiSync Transparency Mode

Added if DLE appears in text.



FULL-DUPLEX PROTOCOLS

On-line applications using CRT terminals required a protocol for full-duplex operations that contained a powerful error detection and correction system that prevented aliasing.

The Binary Synchronous Communications procedures were devised at a time when most data communications circuits were operated at 2400 bits per second, and were two-wire half-duplex circuits connecting remote job entry card reader/printer terminals to mainframes. As on-line applications using remote CRT terminals became more cost-effective, and four-wire private line circuits became available, the demand for a communications protocol that could handle full-duplex operations arose. The requirements for such a protocol were:

1. Messages must flow in both directions simultaneously.
2. It must be possible to have more than one message in the channel at one time (BiSync allows only one).
3. Transparency must be designed in, not tacked on as an afterthought.
4. The protocol must allow switched network, half-duplex, and multipoint operation, as well as full-duplex point-to-point.
5. The error detection and correction scheme must be powerful, and must prohibit the problem of aliasing.

The last condition is a particularly difficult one. Aliasing occurs when a message fragment caused by a transmission error is interpreted as a good message at the receiver; that is, when a bad or broken message "looks like" a good message. This can be a serious problem, especially in critical applications such as funds transfer and military command and control. Much design effort in the international standards organizations and in private companies was devoted to development of protocols that prevented the aliasing problem. This led to the development of three widely used schemes: two of them were alike in principle and operation and in requiring special hardware, but promulgated by different organizations. The third, developed at Digital Equipment Corporation, uses a different technique, but can operate with standard character-oriented line interface equipment.

High-level Data Link Control Procedures

Protocols to prevent aliasing must determine where a true message block begins and ends and what part of the message is to be included in the CRC. HDLC is one of these.

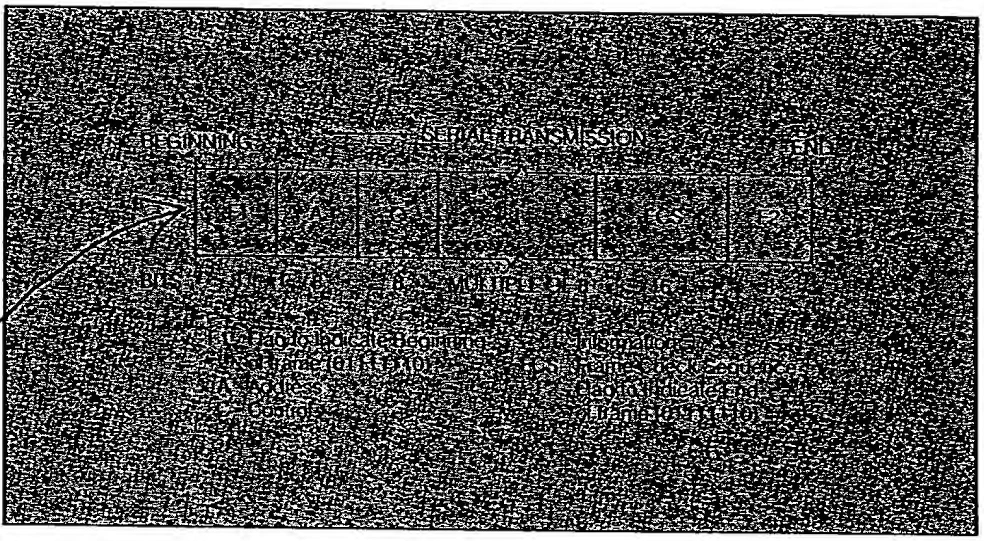
The critical issue in preventing aliasing is the determination of where a legal message block begins and ends, and exactly what part of the information taken in by the receiver is to be included in the CRC. As a result of work done at IBM by J. Ray Kersey and others in the early 1970s, a protocol was proposed and later adopted as an international standard by the International Standards Organization in 1979. It is called the High-level Data Link Control (HDLC) procedures. In HDLC, the message synchronization indicator (called a Flag) is generated by a hardware circuit, and other hardware circuits prevent any data being transmitted from having the same pattern as the Flag. The Flag then becomes a kind of out-of-band framing signal, much like the break signal in the teletypewriter protocol. Since the data being transmitted are examined on a bit-by-bit basis to screen out possible aliases of the Flag; HDLC and other similar protocols, such as IBM's Synchronous Data Link Control (SDLC), are referred to as bit-oriented protocols, or BOPs. Unlike BiSync and DDCMP (which will be described later) the text part of messages sent using the HDLC protocol may, in principle, be any arbitrary number of bits long, and HDLC is defined to allow this. In practice, like BiSync and DDCMP, real implementations of BOPs restrict the text (and in fact all of the message including the Flag) to be an integral multiple of the number of bits in a character (almost always eight).

Data is examined on a bit-by-bit basis in HDLC, thus, it is called a bit-oriented protocol.

In HDLC, all information is carried by frames which may be of three types: information frames (I-frames), supervisory control sequences (S-frames), or unnumbered command/responses (U-frames). *Figure 7-7* shows one information frame as a rectangular block divided into its six fields: a beginning Flag (F1) field, an Address field (A), a Control field (C), an Information field (I), a Frame Check Sequence (FCS) field, and a final Flag (F2) field. S-frames and U-frames have the same fields except the I field is left out.

Figure 7-7.
HDLC Format
(Source: J.L. Fike and G.E. Friend, Understanding Telephone Electronics,*© 1983, Texas Instruments Incorporated)*

Each information frame is divided into six fields.



I-frames perform information transfer, and independently carry message acknowledgements, and Poll or Final bits. S-frames perform link supervisory control such as message acknowledgements, retransmit requests, and requests for temporary holds on I-frame transmissions (like a WACK in BiSync). U-frames provide a flexible format for additional link control data by omitting the frame sequence numbers and thus providing a place for an additional 32 command and 32 response functions. The fields in the HDLC frame are used as follows:

Flag field: Every frame begins and ends with a Flag, which is the bit pattern 01111110. The same Flag may end one frame and begin the next. Every station connected to a link must continuously search the received data for the Flag.

Address field: In command frames, the address identifies the destination station for the command. In response frames, the address identifies the station sending the response.

In HDLC, there are six fields in an information frame, but only five fields in other frames, which are supervisory control sequences or unnumbered command/responses.

Control field: The control field carries commands and responses, according to *Table 7-3*.

Information field: The information field may contain any sequence of bits, which in principle need not be related to a particular character set or data structure. In practice, the information field is almost always an integral multiple of one character in length, which is usually eight bits.

Frame Check Sequence (FCS) field: The FCS (or CRC) for HDLC is the CCITT-CRC as discussed above.

In order to ensure that the Flag is unique, the transmitting hardware must monitor the bit stream continually between the beginning and ending Flags for the presence of strings of five one-bits in a row. If such a string occurs, a zero-bit is inserted (called bit stuffing) by the hardware after the fifth one-bit so the string will not "look like" a Flag. The added zero-bit is removed at the receiver. This procedure makes any appearance of strings of one-bits greater than five in number either a Flag, an error on the transmission line, or a deliberately sent fill pattern between frames of all one-bits. One method of aborting transmission of a frame is to begin transmitting continuous ones.

Specification in the protocol of a unique, hardware-generated Flag pattern and the length of the Address, Control, and FCS fields provides complete transparency for the Information field. The hardware prevents any bit pattern sent by the applications program from having more than five continuous ones, and the hardware-added zeros are stripped from the data stream as it passes through the receiver. Also, the position of the FCS is defined by receipt of the ending Flag, so that the residual pattern in the CRC register can be immediately compared with the fixed value (0001110100001111).

Table 7-3. Control Field Contents

	Bits In Control Field				
HDLC Frame Format	**1**	**2**	**3 4**	**5**	**6 7 8**
I-frame (Information transfer commands/responses)	0	N(S)[1]		P/F[2]	N(R)[3]
S-frame (Supervisory commands/ responses)	1	0	S[4]	P/F	N(R)
U-frame (Unnumbered commands/ responses)	1	1	M[5]	P/F	M

Notes:
1. N(S) is the transmitting station send sequence number. Bit 2 is the low order bit.
2. P/F is the Poll bit for primary station transmissions and the final bit for secondary station transmissions.
3. N(R) is the transmitting station receive sequence number. Bit 6 is the low order bit.
4. S are supervisory function bits.
5. M are modifier function bits.

The order of bit transmission is defined for addresses, commands, responses, and sequence numbers to be low-order bit first, but the order of bit transmission for the information field is not specified. The FCS is transmitted beginning with the coefficient of the highest order term (x^{15}). An invalid frame is defined as one that is not properly bounded by Flags, or is shorter than 32 bits between Flags. Invalid frames are ignored (in contrast with frames that have a bad FCS, which require a NAK).

HDLC Semantics

A normal message sequence involves sending frames from a transmitting station (data source) to a receiving station (data sink), and having the receiving station acknowledge receipt of the transmission by sending a frame back to the sender.

The above discussion dealt with what is called the syntax of HDLC. The syntax is the definition of the bit patterns and order of sending bits that will make correctly formed messages; that is, those that are legal in the protocol. In order to understand what happens when HDLC is used, however, it is necessary to define the semantic content of messages; that is, the meanings that should be assigned to correctly formed messages.

The normal sequence of messages in HDLC consists of the transfer of one or more frames containing I-fields from a data source (the transmitting station) to a data sink (the receiving station). The receipt is acknowledged by the sink by sending a frame in the backwards direction. The source must retain all transmitted messages until they are explicitly acknowledged by the sink. The value of N(R) indicates that the station transmitting the N(R) has correctly received all I-frames numbered up to N(R)-1. I-frames and S-frames (see *Table 7-3* above) are numbered, the number going from 0 to 7 (for unextended control fields). An independent numbering sequence is carried for each data source/data sink combination. The response from a sink may acknowledge several (but not more than 7) received messages at one time, and may be contained in an I-frame being sent from the sink to the source.

A data link consists of two or more communicating stations, so for control purposes it is necessary to designate one station as the primary station, with responsibility for managing data flow and link error recovery procedures. Primary stations send command frames. Other stations on the link are called secondary stations, and they communicate using response frames. Primary stations can send to secondaries using the Select bit in the control field of an I-frame, or a primary may allow a secondary to send by sending a Poll bit. Secondary stations may operate in one of two modes; a Normal Response Mode (NRM) or an Asynchronous Response Mode (ARM). In NRM, the secondary may send only in response to a specific request or permission by the primary station. The secondary explicitly indicates the last frame to be sent by setting the final bit (F) in the control field. In ARM, the secondary may independently initiate transmission without receiving an explicit permission or Poll from the primary.

Space does not permit covering HDLC in greater detail. The interested reader is referred to International Standards ISO 3309-1979 (E), ISO 4335-1979 (E), and ISO 6256-1981 (E) for the complete definition of the HDLC protocol. There is a data link control procedure standard promulgated by the American National Standards Institute (ANSI) in the United States called Advanced Data Communications Control Procedures (ADCCP), which is functionally equivalent to HDLC.

Synchronous Data Link Control

SDLC developed by IBM has the same frame format and is similar in function to HDLC.

The standard full-duplex synchronous data link control protocol used by the IBM Corporation in products conforming to their System Network Architecture is called Synchronous Data Link Control (SDLC). It is functionally equivalent to HDLC, but with these exceptions:

1. SDLC information fields must be an integral multiple of 8 bits long.
2. Current IBM products support only the Unbalanced Normal operation with Normal Disconnected mode class of procedures as defined in ISO DIS 6159 and DIS 4335/DAD1.
3. SDLC contains additional commands and responses not defined in the ISO Elements of Procedure; for example, a TEST command and response. *Figure 7-8* shows the frame structure of SDLC (which is the same as HDLC) along with the common modes, commands, and responses. A close examination of this figure will help you understand the SDLC format and the differences between HDLC and SDLC.

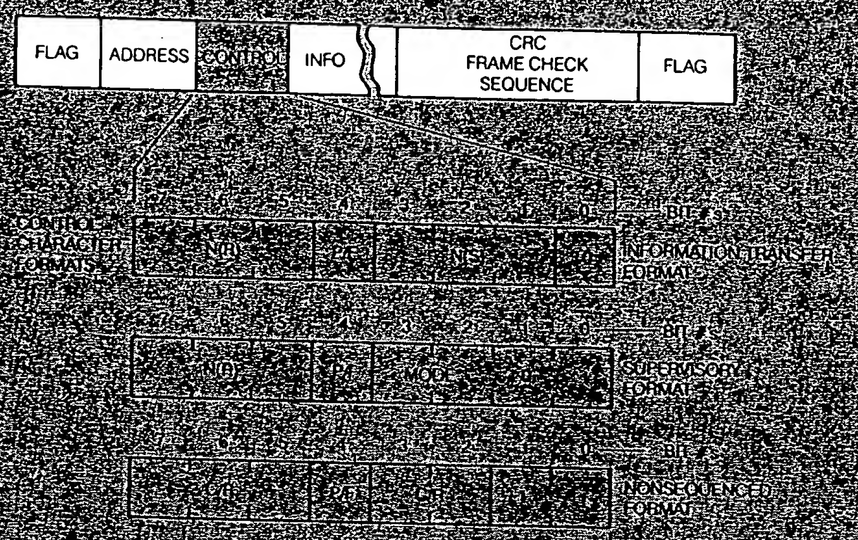
Digital Data Communications Message Protocol

DDCMP is character oriented rather than bit oriented.

At about the same time SDLC was being developed at IBM, George Friend, Steven Russell, and Stuart Wecker were given the task of developing a synchronous protocol at Digital Equipment Corporation. The requirements were similar to those given above for full-duplex protocols, but with one important additional item: the protocol must run on existing data communications hardware, and preferably on asynchronous as well as synchronous links. The result was the Digital Data Communications Message Protocol, or DDCMP.

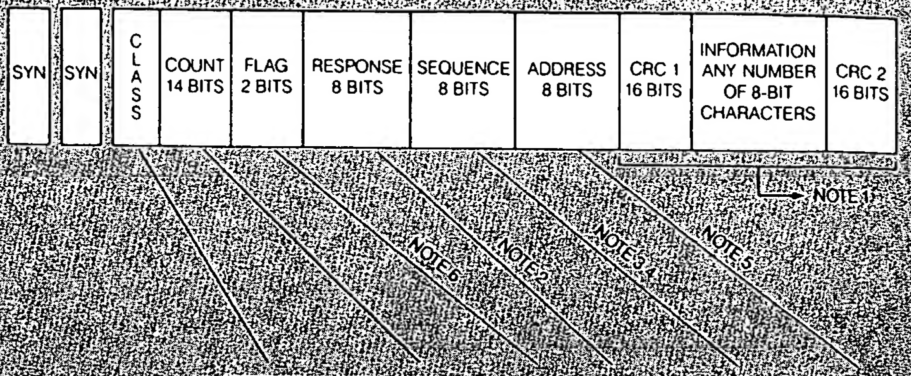
DDCMP is a character oriented protocol, rather than a bit oriented protocol as is SDLC. Therefore, DDCMP requires no special bit stuffing and destuffing hardware, and can be used with various types of line interfaces, including asynchronous units. The method of specifying the length of a message in order to look for the BCC at the right time is inclusion of a 14-bit count field in the header. This is a count of the number of characters in the information field of the message. Since error-free transmission depends on the count field being detected correctly, the header of DDCMP messages is a constant length and has its own BCC, which is checked before setting up to receive the information part of the message. The format of DDCMP messages is shown in *Figure 7-9*. A detailed study of this figure reveals the bit pattern of the various fields in the frame for different types of messages.

Figure 7-8.
SDLC Format
(Source: Kenneth Sherman, Data Communications: A User's Guide, © 1981, Reston Publishing Co., Reston, VA.)



Flag is always 01111110 (If a sequence of 5 1 bits occur in the middle of a transmission the transmitter "stuffs" a "0" bit and the receiver removes it.)
Address is the Station Address on a particular line.
N(S) is Sending Sequence Number — 000 through 111 and then repeats.
P/F is Primary Station Poll when = 1 from Primary Station
Secondary Station Final when = 1 from Station on the line.
N(R) is Receiving Sequence Number — 000 through 111 and then repeats.
Mode is Receive Ready (RR) when 00
Receive Not Ready (RNR) when 10
Reject (REJ) when 01
C/R is Command from Primary Station or Response from Secondary Station.
Nonsequenced Information (NSI) — 000 00
Set Normal Response Mode (SNRM) — 001 00
Disconnect (DISC) — 010 00
Optional Response Poll (ORP) — 100 00
Set Initialization Mode (SIM) — 000 10
} Commands
Nonsequenced Information (UI) — 000 00
Nonsequence Acknowledgment (UA) — 110 00
Request for Initialization (RIM) — 000 10
Command Reject (FRMR) — 001 10
Request Online (DM) — 000 11
} Responses
Info is Information — Variable Length for Information Transfer
— Prohibited for Supervisory Format
— Variable Length for Nonsequenced Format with NSI
— Fixed Format with CMDR
CRC — Cyclic Redundancy Check Remainder

Figure 7-9. DDCMP Message Format in Detail
(Source: John E. MacNamara, Technical Aspects of Data Communication, © 1977, Digital Press, Maynard, Mass.)



	xxxxxxxx	xxxxxxxxxxxxxx	xx	xxxxxxxx	xxxxxxxx	xxxxxxxx
Data Messages	10000001	Character Count	QS	Resp #	Message #	Address
Acknowledgement	00000101	00000001000000	QS	Resp #	00000000	Address
Negative Acknowledge	00000101	00000010REASON	QS	Resp #	00000000	Address
Reason						
BCC Header Error		000001				
BCC Data Error		000010				
Rep Response		000011				
Buffer Unavailable		001000				
Receiver Overrun		001001				
Message Too Long		010000				
Header Format Error		010001				
Reply Message	00000101	00000001100000	QS	00000000	LstMess #	Address
Start Message	00000101	00000110000000	11	00000000	00000000	Address
Start Acknowledgement	00000101	00000111000000	11	00000000	00000000	Address
Maintenance Message	10010000	Character Count	11	00000000	00000000	Address

Notes

1. Only the Data Message and the Maintenance Message have character counts, so only these messages have the information and CRC2 fields shown in the message format diagram above.
2. "RESP #" refers to Response Number. This is the number of the last message received correctly. When used in a negative acknowledge message, it is assumed that the next higher numbered message was not received, was received with errors, or was unaccepted for some other reason. See "Reasons."
3. "Message #" is the sequentially assigned number of this message. Numbers are assigned by the transmitting station modulo 256, i.e., message 000 follows 255.
4. "LstMess #" is the number of the last message transmitted by the station.
5. "Address" is the address of the tributary station in multipoint systems and is used in messages both to and from the tributary. In point-to-point operation, a station sends the address "1" but ignores the address field on reception.
6. "Q" and "S" refer to the quick sync flag bit and the select bit.

The DDCMP does not require any special hardware to perform successfully on synchronous, asynchronous or parallel data channels.

DDCMP has two principal advantages and some disadvantages. One advantage is that it is usable without special hardware on asynchronous, synchronous, or even parallel data channels. Another advantage is that the sequence number field allows for up to 255 messages to be outstanding on the channel at one time, a requirement when operating full-duplex channels over satellite links. On the other hand, it does not support a go-back-N or selective reject mode of operation, and it is, in principle, more vulnerable to the aliasing problem than bit oriented protocols. It also has been mistakenly criticized for inefficiency because of the inclusion of a BCC after the header on the equivalent of I-frames. In fact, for messages of average length, DDCMP is somewhat more efficient than bit-stuffing protocols because of the extra bits added by the hardware in the bit oriented schemes. In practice, aliasing has not caused noticeable problems with DDCMP.

WHAT HAVE WE LEARNED?

1. Protocols are rules for communications between processes that are alike. Interfaces are rules for communicating between processes that are different.
2. Data communications protocols are made up of symbols to be communicated, a code set translating those symbols to a binary code, and rules for the correct sequencing of the symbols.
3. Parity is a method of adding redundancy to coded symbols to help detect errors in transmission.
4. XMODEM is a simple protocol used in asynchronous transmission between microcomputers.
5. A Cyclic Redundancy Check is a number calculated from the data transmitted by the sender and recalculated by the receiver that allows the data to be checked for errors in transmission.
6. The most widely used code sets in the U.S.A. are ASCII and EBCDIC.
7. Protocols must be suitable for use on dial and private line connections, and for point-to-point and multipoint network arrangements.
8. The most widely used protocols in the U.S.A. are TTY, BiSync, SDLC, and DDCMP.

B

MOTOROLA

Microprocessor and Memory Technologies Group

Order this document by MC68EN302/D

MC68EN302

Product Brief

Integrated Multiprotocol Processor with Ethernet

Motorola introduces a version of the well-known MC68302 Integrated Multiprotocol Processor (IMP) with Ethernet and DRAM controllers. It is known as the MC68EN302, and expands a family of devices based on the MC68302.

The Ethernet controller has a 16-bit interface, resides on the 68000 bus and provides complete IEEE 802.3 compatibility. The programming model is adopted from the standard 68302 programming model. The DRAM controller is adopted from the MC68306 product. It is enhanced to support both parity and external bus masters.

The MC68EN302 is packed in a low profile 144 TQFP.

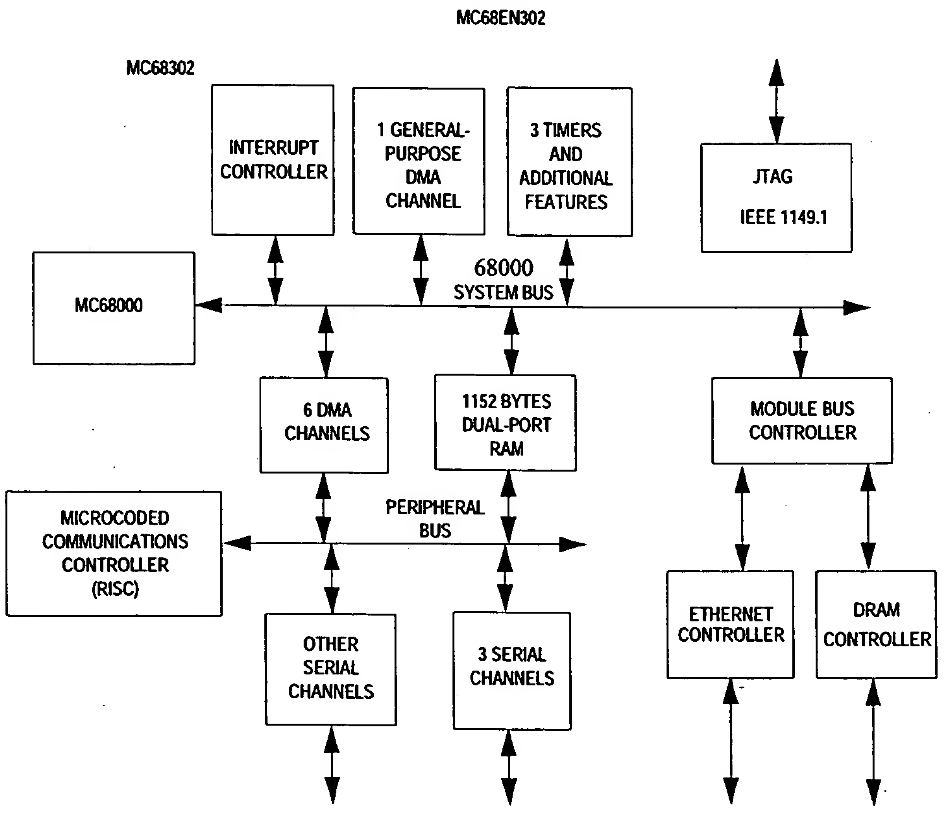


Figure 1. MC68EN302 Block Diagram

This document contains information on a product under development. Motorola reserves the right to change or discontinue this product without notice.

SEMICONDUCTOR PRODUCT INFORMATION

FEATURE LIST

The following features are incorporated into the MC68EN302 device:

- Full Complement of Existing Three SCC's Plus Ethernet Channel
- Ethernet Channel Fully Compliant with IEEE 802.3 Specification.
- Supports Data Rates up to 10 Mbps.
- Supports the "68302" Style Programming Model.
- On-Chip Descriptors Lower Processor Bus Bandwidth Requirements.
- Separate 128 Byte FIFOs for Transmit and Receive.
- Automatic Internal Retransmission (which Frees the Processor Bus).
- Automatic Internal Flushing of Receive FIFO During Collisions (which Frees the Processor Bus).
- Dynamic Bus Sizing Support for 8-Bit Devices
- Glueless Dynamic RAM Controller without External Bus Master
- Address Muxing Support for External Bus Masters Using DRAM Controller
- Fully IEEE 1149.1 JTAG Compliant
- 144 TQFP Package for Up to 25 MHz

ETHERNET CONTROLLER

The Ethernet controller consists of a Ethernet protocol core, transmit and receive FIFOs, and a 16-bit wide data/control interface to a 68000 bus (refer to Figure 2). The Ethernet protocol core (EPC) provides compatibility with the IEEE 802.3 Ethernet standard. The transmit and receive FIFOs allow automatic handling of collisions and collision fragments by the EPC, and they also provide for bus latency that can be encountered by the DMA channels. Separate DMA channels are used for transmit and receive data paths. A dual-port RAM is used for the on-chip buffer descriptors. A buffer descriptor control (BDC) block updates the buffer descriptors. Control status registers are used for direct control of all of the blocks in the Ethernet controller.

ETHERNET FEATURES

- Does Not Affect Performance of Existing SCCs
- 802.3 MAC Layer Support
- Compatible with 68160 EEST (Twisted Pair/AUI)
- Two Dedicated Ethernet DMA channels, Transmit and Receive
- Full-Duplex (Switched) Ethernet Support
- Up to 10 Mb/s Operation (20 Mb/s Full-Duplex)
- 128-Byte FIFO on both Transmit and Receive
- No CPU or Bus Overhead Required on Rx or Tx Frame Collisions
- 64 entry CAM with Hash Option
- 128 internal Buffer Descriptors
- Performs Framing Functions

- Full Collision Support
- Receives Back-to-Back Frames
- Detection of Receive Frames That Are Too Long
- Multi-Buffer Data Structure
- Supports 48-Bit Addressing
- Heartbeat Indication
- Transmitter Network Management and Diagnostics
- Receiver Network Management and Diagnostics
- Loopback Mode for Testing
- Non-Aggressive Deferral Option
- Heartbeat Status and Interrupt Option
- Graceful Stop Command

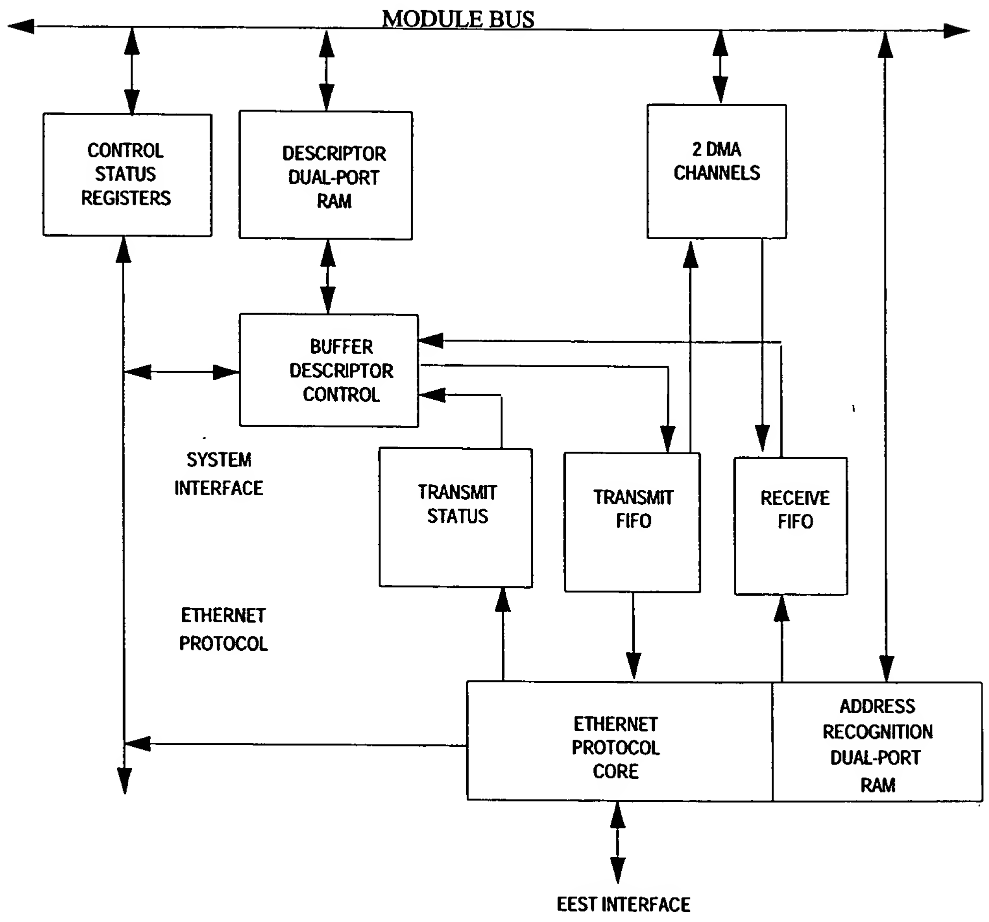


Figure 2. Ethernet Controller Block Diagram

MODULE BUS CONTROLLER

The MC68EN302 module bus controller provides basic interface capabilities to the module bus as well as basic system responsibilities. The features of the module bus controller are:

- Interface Between Internal 68000 bus and the Module Bus.
- Provision for Dynamic Bus Sizing, Using the Chip Select Logic of the 68302 Core.
- Handling of Interrupts for the Ethernet Controller and DRAM Controller.
- Coordination of Bus Mastership from External Sources, the Module Bus, and the MC68EN302 Core.

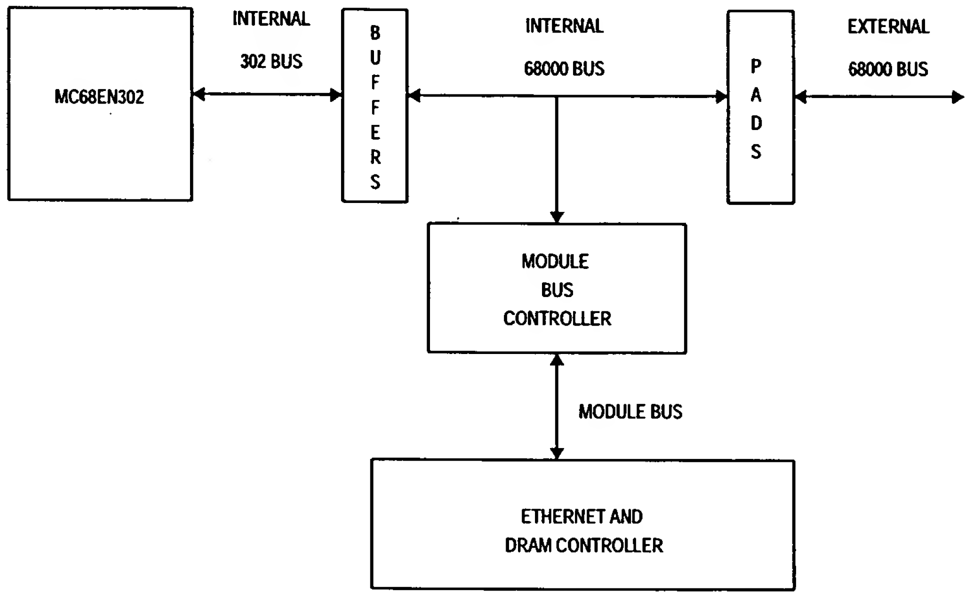


Figure 3. BusStructure

DRAM CONTROLLER

- Provides two CAS lines
- Provides two RAS lines (two banks supported)
- DRAM address multiplexing on standard address bus
- Programmable up to three wait states
- 100 nS DRAM for zero wait states at 20 MHz
- 80 nS DRAM for zero wait states at 25 MHz
- CAS before RAS refresh and refresh support during system reset

- Programmable refresh period and pre-charge period
- RAS lines are separate from the four chip selects
- Refresh hidden from bus accesses
- Write protect option
- Each bank programmable size from 128Kbytes to 8Mbytes

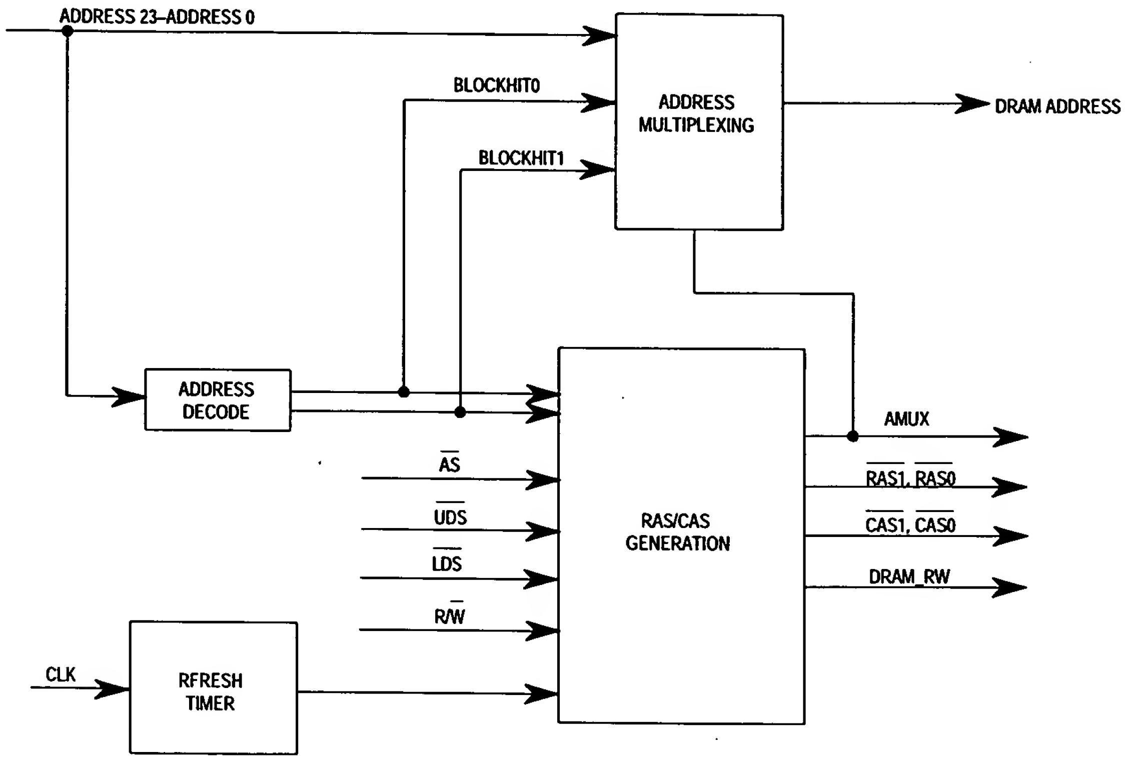


Figure 4. DRAM Controller

MC68EN302 APPLICATIONS

The MC68EN302 is intended for low-end bridge and router applications. It has the three SCCs from the MC68302, plus an additional Ethernet interface giving it a total of four serial interfaces.

Since the MC68EN302 has both the three SCCs as well as an Ethernet interface, it would be an excellent choice in an ISDN to Ethernet router.

For remote access dial-in, the MC68EN302 could be used in a dial-up modem that would connect to an Ethernet LAN.

- Low End Bridges

- Industrial Control
- Remote LAN Access Points for Remote Dial-In
- PCMCIA Ethernet + WAN Cards
- Communication System Control Boards
- Intelligent Peripheral Chip to an 020/030

MC68EN302 PIN DESCRIPTION

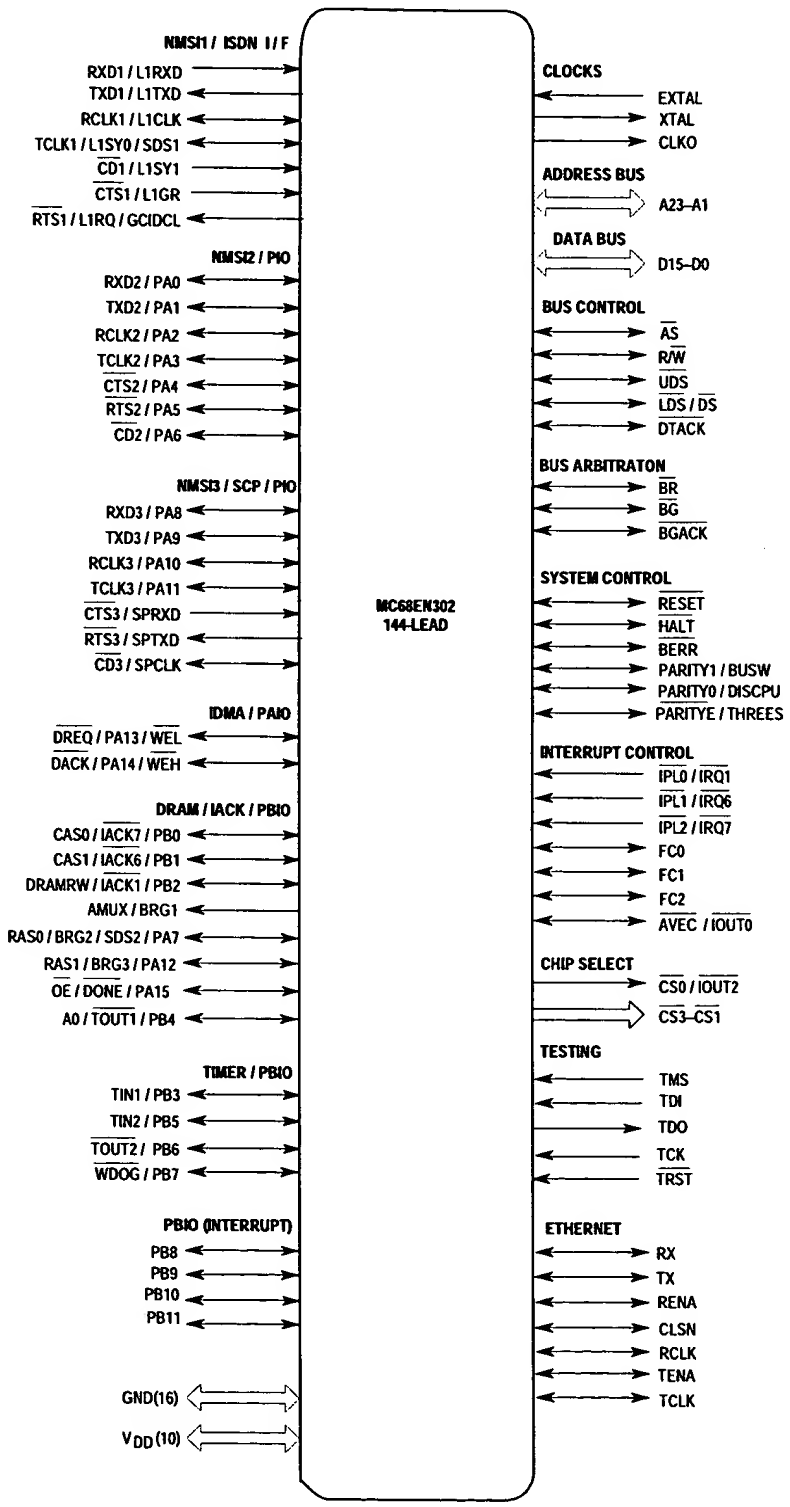


Table 1. MC68EN302 Ordering Information

Package Type	Operating Voltage	Frequency (MHz)	Temperature	Order Number
Thin Quad Flat Pack (PV Suffix)	5V	20	0°C to 70°C	MC68EN302PV20
Thin Quad Flat Pack (PV Suffix)	5V	25	0°C to 70°C	MC68EN302PV25

Table 2. Documentation

Document Title	Order Number	Contents
MC68302 User's Manual	MC68302UM/AD	Detailed information for design
M68000 Family Programmer's Reference Manual	M68000PM/AD	M68000 Family Instruction Set
The 68K Source	BR729/D	Independent vendor listing supporting software and development tools
The MC68EN302 Addendum		Describes the differences between the MC68302 and the MC68EN302

Literature Distribution Centers:

USA: Motorola Literature Distribution; P.O. Box 20912, Arizona 85036.
EUROPE: Motorola Ltd.; European Literature Centre; 88 Tanners Drive, Blakelands, Milton Keynes, MK14 5BP, England.
JAPAN: Nippon Motorola Ltd.; 4-32-1, Nishi-Gotanda, Shinagawa-ku, Tokyo 141 Japan.
ASIA-PACIFIC: Motorola Semiconductors H.K. Ltd.; Silicon Harbour Center, No. 2 Dai King Street, Tai Po Industrial Estate, Tai Po, N.T., Hong Kong.

C

Oki Semiconductor

▫ Home ▫ Buy Samples ▫ Sales ▫ Jobs ▫ Procurement ▫ Other

the Right Partner for a Digital World™

DOCUMENTATION SUPPORT PRESS COMPANY

Search for

GO

About Search

Oki Products
- ARM® Solutions
- ASICs/SOC
- MCUs/MPUs
- Connectivity ICs
- Telecom ICs
- Speech ICs
- Display Drivers
- DRAMs/ASMs
- ROMs

Applications
- Packaging
  - W-CSP NEW!
- Oki Optical Components

Products » Telecom ICs » PCM A- / µ-Law CODECs

ML7074-001GA

Speech CODEC for VoIP

Documents Applications Description Block Diagram Features

Documents

» A PDF of this Data Sheet (471 KB) is available. **ML7074_001GA.pdf**

Applications

» **Line Cards**

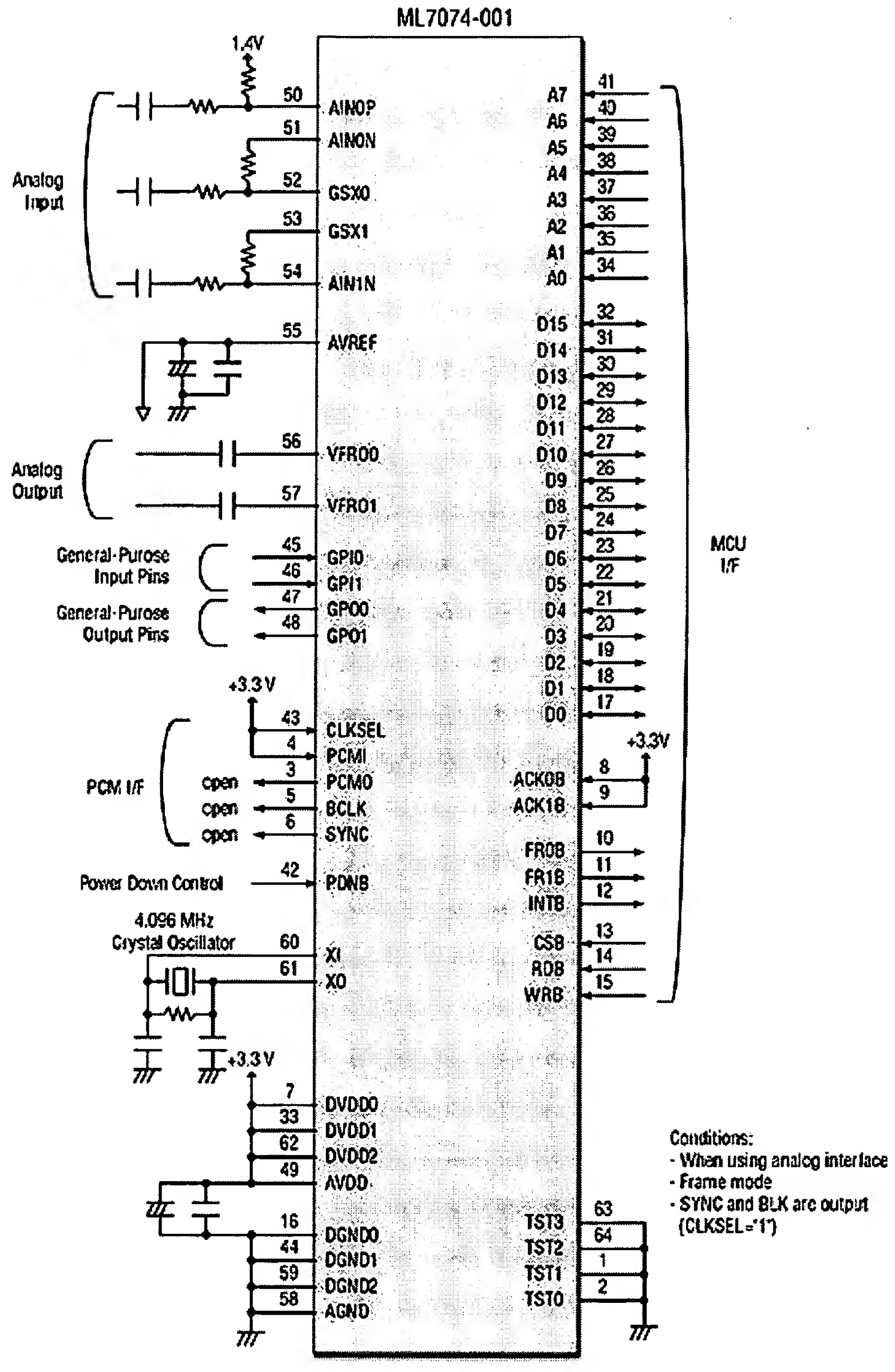
Description

Oki Semiconductor's ML7074-001GA is a speech CODEC designed for VoIP applications.
conformance selection for various VoIP standard environments including G.729.A (8 kbp
G.711 (64 kbps) µ-law, and A-law, and a mutual conversion function between G.729.A a

The ML7074-001GA is optimized for adding VoIP functions to terminal adapters, routers
rich feature set that includes an echo canceller for 32 msec delay, DTMF detection, two
and tone generation, built-in FIFOs, I/P and O/P amplifiers. This CODEC uses a single 3.
a 64-pin plastic QFP (QFP64-P-1414-0.80-BK) package.

BACK TO TOP

Block Diagram



BACK TO TOP

Features

- Single 3.3-V power supply operation (DVDD 0, 1, 2, AVDD : 3.0 to 3.6 V)
- Speech CODEC:
  - Selectable among G.729.A (8 kbps), G.726 (32 kbps), G.711 (64 kbps) μ-law, ar
  - Mutual conversion function between G.729.A (8 kbps) and G.726 (32 kbps).
- Echo canceller for 32 ms delay

- DTMF detection function
- Tone detection function: 2 systems (1650 Hz, 2100 Hz: Detect frequency can be cha
- Tone generation function
- FSK generation function
- Dial pulse detection function
- Dial pulse transmit function
- Internal 1-channel 16-bit timer
- Built-in FIFO buffers (640 bytes) for transferring transmit and receive data:
  - Frame / DMA (slave) interface selectable.
- Master clock frequency: 4.096 MHz (crystal oscillation or external input)
- Hardware or software power-down operation option
- Analog input / output type:
  - Two built-in input amplifiers, 10 k driving
  - Two built-in output amplifiers, 10 k driving
- Package:
  - 64-pin plastic QFP (QFP64)

BACK TO TOP

Applications

- VoIP applications - G.729.A, G.726, G.711 μ-law, and A-law
- Terminal adapters
- Routers and gateways
- I / P-phones

home : documentation : company : press : sales : jobs : support : packaging : buy samples
privacy policy : copyright : other oki sites

Oki Semiconductor, Serving North and South America